

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-257687

#1

(43)Date of publication of application : 08.10.1993

(51)Int.Cl.

G06F 9/38

G06F 9/38

G06F 9/45

(21)Application number : 04-347929

(71)Applicant : HITACHI LTD

(22)Date of filing : 28.12.1992

(72)Inventor : HOTTA TAKASHI
NAKATSUKA YASUHIRO
TANAKA SHIGEYA
YAMADA HIROMICHI
MAEJIMA HIDEO

(30)Priority

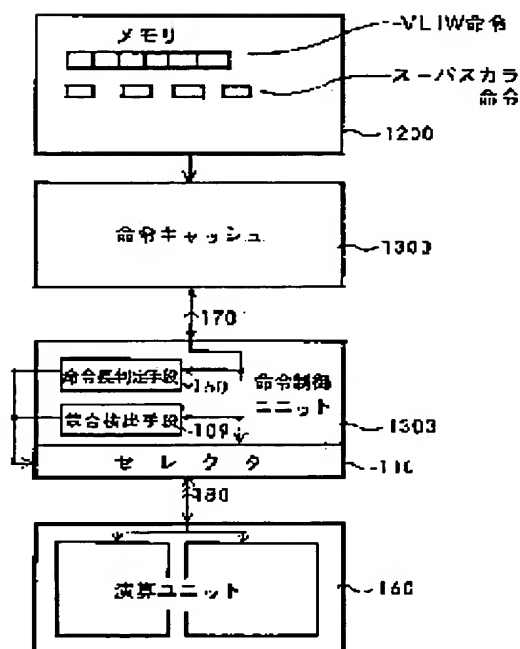
Priority number : 04 29 Priority date : 06.01.1992 Priority country : JP

(54) COMPUTER HAVING PARALLEL ARITHMETIC FUNCTION

(57)Abstract:

PURPOSE: To execute a parallel processing of an operation and to improve the performance by executing plural pieces of instructions of short instruction length for instructing a single operation of one machine cycle, or one piece of instruction of long instruction length for instructing plural operations.

CONSTITUTION: An instruction control unit 1303 reads out an instruction from an instruction cache by using an interface 170, decodes it, and controls an arithmetic unit 160 through an interface 180. The arithmetic unit 160 can process in parallel plural operations. In this case, the unit has a 4-byte length instruction for instructing a single operation, and a 16-byte length instruction for instructing plural operations, and in an instruction cache 1300, the 16-byte length instruction and the 4-byte length instruction are mixed and placed so that there is no competition between the 16-byte length instructions, and between the 16-byte length instruction and the 4-byte length instruction. A competition detecting means 109 detects a competition only between the 4-byte length instructions. In accordance with an output of the competition detecting means 109, a selector 110 selects an operation which can be executed in parallel and decodes it.



LEGAL STATUS

[Date of request for examination] 31.03.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3146707

[Date of registration] 12.01.2001

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision
of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office



PatentWeb
Home



Edit
Search



Return to
Patent List



Back to
Record



Help

MicroPatent® Worldwide PatSearch: Record 3 of 3

#1

Family of JP05257687 [How It Works](#)

Stage 2 Patent Family - "Extended"		Priorities and Applications	
CC DocNum	KD PubDate	CC AppNum	KD AppDate
<input type="checkbox"/> CN 1034604	B 19970416	CN 94101921 KR 9300074	A 19940106 A 19930106
<input type="checkbox"/> CN 1094135	A 19941026	CN 94101921 KR 9300074	A 19940106 A 19930106
<input type="checkbox"/> DE 69325826	CO 19990909	DE 69325826 JP 29	A 19930105 A 19920106
<input type="checkbox"/> DE 69325826	T2 20000224	DE 69325826 JP 29	A 19930105 A 19920106
<input type="checkbox"/> EP 551090	A2 19930714	EP 93100067 JP 29	A 19930105 A 19920106
<input type="checkbox"/> EP 551090	B1 19990804	EP 93100067 JP 29	A 19930105 A 19920106
<input type="checkbox"/> EP 551090	A3 19940907	EP 93100067 JP 29	A 19930105 A 19920106
<input type="checkbox"/> EP 855647	A1 19980729	EP 93100067 EP 98103921 JP 29	A3 19930105 A 19930105 A 19920106
<input type="checkbox"/> JP 3146707	B2 20010319	JP 29 JP 347929 JP 347929	A 19920106 A 19921228 A 19921228
<input type="checkbox"/> JP 5257687	A2 19931008	JP 29 JP 347929	A1 19920106 A 19921228
<input type="checkbox"/> KR 259738	B1 20000701	JP 29 KR 9300074	A 19920106 A 19930106
<input checked="" type="checkbox"/> US 5503540	A 19960402	KR 9300074 US 177972	A 19930106 A 19940106
<input type="checkbox"/> US 5680637	A 19971021	JP 29 US 877 US 599856 US 599856	A 19920106 B1 19930105 A 19960213 A 19960213
13 Publications found.			

Order Selected Documents

United States Patent [19]

Hotta et al.

[11] Patent Number: **5,680,637**[45] Date of Patent: **Oct. 21, 1997****[54] COMPUTER HAVING A PARALLEL OPERATING CAPABILITY**

[75] Inventors: **Takashi Hotta; Yasuhiro Nakatsuka; Shigeya Tanaka; Hiromichi Yamada; Hideo Maejima**, all of Hitachi, Japan

[73] Assignee: **Hitachi, Ltd.**, Tokyo, Japan

[21] Appl. No.: **599,856**

[22] Filed: **Feb. 13, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 877, Jan. 5, 1993, abandoned.

[30] Foreign Application Priority Data

Jan. 6, 1992 [JP] Japan 4-000029

[51] Int. Cl.⁶ **G06F 9/30**

[52] U.S. Cl. **395/800; 395/391; 395/392; 395/384; 395/382**

[58] Field of Search **395/800, 375, 395/397, 384, 382, 391**

[56] References Cited**U.S. PATENT DOCUMENTS**

5,136,691 8/1992 Baror 395/200
5,163,139 11/1992 Haugh et al. 395/375

FOREIGN PATENT DOCUMENTS

2-130634 of 1990 Japan .

OTHER PUBLICATIONS

Johnson, William *Superscalar Processor Design*, 1991 Prentice Hall, Inc. pp. 5, 6, 25, 105-106, 177, 180-181, 240.
Cohn, R. et al, "Architecture and Compiler Tradeoffs for a Long Instruction Word Microprocessor", 3rd International Conference on Architectural Support for Programming Languages and Operating Systems, 1989, pp. 2-14.

Yoshida et al. (1991) "The Approach to Multiple Instruction Execution in Gmicro/400 Processor" (185-195).

Computer Architecture News, "Software Prefetching", D. Callahan, et al., Apr. 19, 1991, No. 2, New York, New York.
Computer Architecture News, "An Architecture for Software-Controlled Data Prefetching", A.C. Klaiber, et al., May 19, 1991, No. 3, 18th Annual Int. on Computer Architecture, New York, New York.

Proceedings Advanced Computer Technology, Reliable Systems and Applications, "HARP: A VLIW RISC Processor", P.A. Findlay, et al., 5th Annual Computer Conference, Bologna, May 13-16, 1991.

Computer Architecture A Quantitative Approach, "Advanced Pipelining-Taking Advantage of More Instruction-Level Parallelism".

"A Variable Instruction Stream Extension to the VLIW Architecture", A. Wolfe, et al.

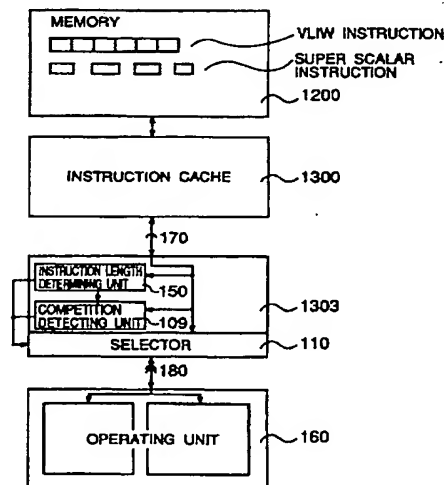
Primary Examiner—Larry D. Donaghue

Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus, LLP.

[57] ABSTRACT

A RISC processor is arranged to reduce a code size, make the hardware less complicated, execute a plurality of operations for one machine cycle, and enhance the performance. The processor is capable of executing N instruction each having a short word length for indicating a single operation or an instruction having a long word length for indicating M (N<M) operations. When the number of operations to be executed in parallel is large, the long-word instruction is used. When it is small, the short-word instruction is used. A competition between the long-word instructions is detected by hardware and a competition between the short-word instructions only is detected by software. The simplification of the hardware brings about improvement of a machine cycle, improvement of a code cache hit ratio caused by the reduction of a code size and increase of the number of operations to be executed in parallel for the purpose of enhancing the performance.

7 Claims, 36 Drawing Sheets



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平5-257687

(43) 公開日 平成5年(1993)10月8日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/38	3 7 0 X	9290-5B		
	3 1 0 X	9290-5B		
9/45		9292-5B	G 0 6 F 9/44	3 2 2 F

審査請求 未請求 請求項の数17(全 33 頁)

(21) 出願番号 特願平4-347929

(22) 出願日 平成4年(1992)12月28日

(31) 優先権主張番号 特願平4-29

(32) 優先日 平4(1992)1月6日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 堀田 多加志

茨城県日立市久慈町4026番地 株式会社日

立製作所日立研究所内

(72) 発明者 中塚 康弘

茨城県日立市久慈町4026番地 株式会社日

立製作所日立研究所内

(72) 発明者 田中 成弥

茨城県日立市久慈町4026番地 株式会社日

立製作所日立研究所内

(74) 代理人 弁理士 小川 勝男

最終頁に続く

(54) 【発明の名称】 並列演算機能を有する計算機

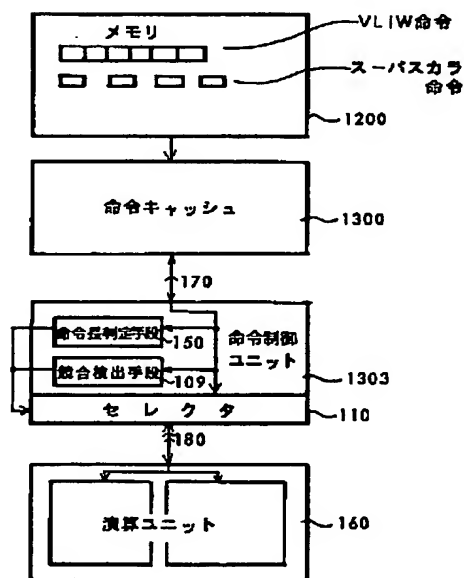
(57) 【要約】

【目的】 R I S プロセッサにおいて、コードサイズを小さく、またハードウェアを複雑にせず、1マシンサイクルに複数の演算を実行し、性能を高める。

【構成】 単一演算を指示する命令長の短い命令をN個実行するか、M個(N<M)の演算を指示する命令長の長い命令を実行することを可能にし、並列に実行できる演算数が多い時には長い命令を、少の時には短い命令を用いる。また、長い命令間の競合はソフトウェアで検出し、短い命令間の競合のみをハードウェアで検出する。

【効果】 ハードウェア容易化によるマシンサイクルの向上、コードサイズ縮小によるコードキャッシュヒット率向上、及び並列に実行できる演算数の向上により、性能が高められる。

図 1



1

【特許請求の範囲】

【請求項1】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納されている命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記プログラムカウンタで指示された上記命令が命令長の長い命令であれば上記レジスタに上記命令を設定し、上記プログラムカウンタで指示された上記命令が命令長の短い命令であれば所定のレジスタに上記命令を設定する命令選択手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項2】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると上記レジスタに上記命令を設定し、上記命令長の短い命令と判定され、かつ、上記競合検出手段によって競合がないと判定されると所定のレジスタに上記命令を設定する命令選択手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項3】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記競合検出手段の内容に応じて上記命令長の短い命令を1マシンサ

2

イクルに所定の数実行し、上記命令長の長い命令と判定されると、命令長の長い命令を1マシンサイクルに所定の数実行する演算手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項4】請求項1、2又は3において、上記命令は、並列して実行できる演算の多い時には上記命令長の長い命令を、並列して実行できる演算の少ない時には上記命令長の短い命令であることを特徴とする並列演算機能を有する計算機。

10 【請求項5】請求項1、2、3又は4において、上記命令長の長い命令と上記命令長の短い命令との使い分けをコンパイラで生成した上記命令を実行することを特徴とする並列演算機能を有する計算機。

【請求項6】請求項1、2又は3において、上記命令長の長い命令の中で指定できる演算数は、1マシンサイクルで処理される上記命令長の短い命令の数よりも大きいことを特徴とする並列演算機能を有する計算機。

20 【請求項7】請求項1、2又は3において、命令長の長い命令と、それ以前の命令長の長い命令との競合が無いようにコンパイラで命令列を生成し、上記コンパイラの出力に基づいて処理を行うことを特徴とする並列演算機能を有する計算機。

【請求項8】請求項1、2又は3において、有効な命令長の短い命令実行後有効な命令長の長い命令を実行する時、及び、有効な命令長の長い命令の実行後有効な命令長の短い命令を実行する時には、両者の間に一定数の無効命令を挿入することを特徴とする並列演算機能を有する計算機。

30 【請求項9】請求項1、2又は3において、命令長の長い命令は、次命令以降に続く無効命令の数を任意に指定できることを特徴とする並列演算機能を有する計算機。

【請求項10】請求項1、2又は3において、命令長の長い命令の演算結果は、以後の任意の命令から反映することを特徴とする並列演算機能を有する計算機。

40 【請求項11】請求項10において、演算結果を反映する一定数後の命令の一定数をN、パイプライン段数をMとした時に、 $N \geq M$ とすることを特徴とする並列演算機能を有する計算機。

【請求項12】請求項10において、過去のレジスタの内容を一定サイクル分保存しておく記憶手段を有することを特徴とする並列演算機能を有する計算機。

50 【請求項13】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

命令長の長い命令の演算結果は、以後の任意の命令から反映することを特徴とする並列演算機能を有する計算機。

【請求項14】請求項13において、過去のレジスタの内容を一定サイクル分保存しておく記憶手段を有することを特徴とした並列演算機能を有する計算機。

【請求項15】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令には、単一演算を指示する命令長の短い命令と複数演算を指示する命令長の長い命令とを有し、上記プログラムカウンタで指示された命令が上記命令長の短い命令か命令長の長い命令かを判定する手段と、上記判定手段の結果が上記命令長の短い命令なら1マシンサイクル中に任意の数の命令を実行し、上記命令長の長い命令なら1マシンサイクル中に1個の命令を実行する手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項16】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令には、単一演算を指示する命令長の短い命令と複数演算を指示する命令長の長い命令とを有し、上記プログラムカウンタで指示された命令が上記命令長の短い命令か命令長の長い命令かを判定する命令長判定手段と、上記命令長の短い命令間の競合を検出する競合検出手段と、上記判定手段の結果が上記命令長の長い命令なら1マシンサイクル中に1個の命令を実行し、上記命令長の短い命令なら上記競合検出手段の結果に応じて、競合が検出されれば、競合が解消されるまで待ってから命令を任意の数実行し、競合が検出されなければ命令を任意の数実行することを特徴とする並列演算機能を有する計算機。

【請求項17】レジスタとメモリとキャッシュメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は、上記メモリまたは上記キャッシュメモリから上記レジスタへデータ転送を指示する第1のフィールドと、上記メモリから上記キャッシュメモリへデータ転送を指示する第2のフィールドを有することを特徴とする並列

演算機能を有する計算機。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、並列演算を実行する計算機に係り、特にスーバスカラ方式とVLIW方式とを混在させて実行する並列演算機能を有する計算機に関する。

【0002】

【従来の技術】計算機アーキテクチャは、半導体技術の進歩等に支えられ、年々進歩している。1980年代には、これまでの複雑な命令をマイクロ命令を使って複数サイクルにかけて処理するCISC(Complex Instruction Set Computer)に代って、簡単な命令を1サイクルで実行するRISC(Reduced Instruction Set Computer)が現れた。

【0003】さらに、演算方式の高速化技術として、スーバスカラ方式とVLIW(VeryLong Instruction Word)方式が提案されている。

【0004】スーバスカラ方式とは、命令実行時にハードウェアで命令間の競合を調べ、競合が無ければ1マシンサイクルに複数命令を実行する方式で、特願昭63-283673号(従来技術1)や、J-Hennessy and D.A Patterson "Computer Architecture AQuantitative Approach" Morgan Kaufmann Publishers, Inc 1990. P. 318 (従来技術2)に記載されている。

【0005】またVLIW方式とは、複数演算器の動作を制御するフィールドを持った長い命令を用いる方式である。通常のRISCプロセッサの命令長が32bitなのに対し、64、128、256以上といった長さの命令を持つ。この方式についての説明も、前記J-Hennessy and D.A Patterson(従来技術1)による文献に記載されている。

【0006】VLIW方式の改良技術として、1語長命令と3語長命令を混在させて、VLIW方式で処理することにより、コードサイズの大きさを改善する技術が、Robert Cohn et al. "Architecture and Compiler Tradeoffs a Long Instruction WordMicroprocessor" Third International Conference on Architectural Support for Programming Languages and Operating System, 1989, P. 2-14 (従来技術3)に記載されている。

【0007】

【発明が解決しようとする課題】以下に、スーバスカラ方式とVLIW方式の特徴について述べる。

【0008】スーバスカラ方式の利点は、単一演算を指示する命令長の短い命令で、有効演算のみを指示するためコードサイズが小さくできる。

【0009】命令を追加する必要がないので前機種との互換性が保たれることである。

【0010】これに対して、スーバスカラ方式の第1の問題点は、並列実行する演算内の競合を検出しないことではないことである。並列演算する演算の数が多くなれば

5

ばなる程、競合検出に要するハードウェア量は大となる。

【0011】また、第2の問題点は、現サイクル以前に実行した命令と現サイクルに実行する命令との間の競合検出、待合わせが複雑であることである。並列演算する演算の数が多くなればなる程、現サイクルの命令と競合する可能性のある命令が多くなり、第2の問題点である両者の競合検出、待合わせのハードウェアは複雑になる。

【0012】また、第3の問題点は命令長が短いため、命令によって指定できるレジスタの数が少ないことである。16〜32本が典型例である。J-Hennessy and D. A Pattersonの文献のP. 325に示されているように、並列して実行可能な演算を増やすためのソフトウェア上の工夫として、ループアンローリングやソフトウェアパイプラインを用いようとすると、レジスタの数が不足する。逆に言えば、存在するレジスタの範囲でしか最適化できない。

【0013】この改善策として、上記従来技術1の文献のE-21〜22に、演算結果を次の命令にすぐには反映させないようにすることで、レジスタの数の不足を改善することができる記載されている。

【0014】また、David Callahan et al. "Software Prefetching" Fourth International Conference on Architectural Support for Programming Languages and Operating System, 1991, P. 40〜52の文献に、スーパスカラマシンにおいて命令によりメインメモリからキャッシュメモリにデータをプリフェッチすることが記載されている。

【0015】以上から、スーパスカラ方式では、命令実行の並列度を増すと上記第1、第2の問題点である競合検出の複雑さからマシンサイクルを高めることができず、処理速度が向上しないという問題を有している。

【0016】次に、VLIW方式の第1の利点は、命令長が長く、1命令の中に複数の演算が指定でき、かつ、命令内での演算間の競合がないため、実行時にハードウェアで、並列実行する演算間の競合を検出しなくてもよいことである。

【0017】第2の利点は、命令長が長いこと、多くのレジスタが指定可能なことである。次にVLIW方式の第1の問題点は、前述の命令内での演算間の競合を避けるため、全てのフィールドに有効な演算を指定できるとは限らず、コードサイズが大きくなってしまふことである。

【0018】第2の問題点は現サイクル以前に実行した命令と現サイクルに実行する命令との間の競合検出、待合わせが複雑なことである。これは、スーパスカラ方式の第2の問題点と同じである。

【0019】これについて、ハードウェアでは競合検出を行わず、コンパイラによって予め競合回避を行う技術

6

が、Andrew Wolf and John P. Shen "A Variable InstructionStream Extension to the VLIW Architecture", Fourth International Conference on Architectural Support for Programming Languages and Operating System, 1991, P. 2〜14に記載されている。また、VLIW方式の第3の問題点は、前機種との互換性が取れないことである。これは、スーパスカラ方式が、従来の1語長命令をハードウェアで並列実行するのに対し、VLIW方式では、命令の再定義が必要となるからである。

【0020】これまでに述べてきたように、スーパスカラ方式とVLIW方式の利点を活かしながら、スーパスカラ方式とVLIW方式の欠点を補う計算機は存在しなかった。

【0021】本発明の第1の目的は、スーパスカラ方式とVLIW方式を混在させて演算実行可能な計算機を提供することにある。これは、単一演算を指示する命令長の短い命令よりなる従来アーキテクチャを持つ計算機の上位互換性を保ちながら、処理速度を向上させることである。

【0022】

【課題を解決するための手段】上記目的を達成するために、本発明によれば、第1に、レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納されている命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記プログラムカウンタで指示された上記命令が命令長の長い命令であれば上記レジスタに上記命令を設定し、上記プログラムカウンタで指示された上記命令が命令長の短い命令であれば所定のレジスタに上記命令を設定する命令選択手段とを有する。

【0023】本発明の第2の特徴によれば、レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると上記レジスタに上記命令を設定し、上記命令長の

7

短い命令と判定され、かつ、上記競合検出手段によって競合がないと判定されると所定のレジスタに上記命令を設定する命令選択手段とを有する。

【0024】本発明の第3の特徴によれば、レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記競合検出手段の内容に応じて上記命令長の短い命令を1マシンサイクルに所定の数実行し、上記命令長の長い命令と判定されると、命令長の長い命令を1マシンサイクルに所定の数実行する演算手段とを有する。

【0025】

【作用】本発明によれば、1マシンサイクルに単一演算を指示する命令長の短い命令を複数個、あるいは、複数演算を指示する命令長の長い命令を1個実行できるので、演算が並列処理され性能が高められる。

【0026】本発明の一態様によれば、並列して実行できる演算の多い時のみ命令長の長い命令を用いることにより、命令長の長い命令の中の無効フィールドを少なくすることができ、コードサイズを小さくすることができる。これにより、主メモリ及びキャッシュメモリの使用効率上がり、処理速度の向上が図れる。

【0027】本発明の他の一態様によれば、命令長の長い命令の中で指定する複数の演算内の競合はあり得ず、ハードウェアでこれを検出する必要はない。ハードウェアは同一サイクルに実行する命令長の短い命令間の競合のみを検出すればよい。本発明によれば、1マシンサイクルで実行される命令長の短い命令の数を、命令長の長い命令の中に指示される演算の数より小さくすることにより、平均的に1マシンサイクルに実行される演算数が高い割に、並列に実行する演算間の競合検出を容易にすることができる。

【0028】本発明の他の一態様によれば、命令長の長い命令と、それ以前の命令長の長い命令との競合が無いようにコンパイラで命令列を生成することが可能であり、ハードウェアでこれを検出する必要はない。

【0029】本発明の他の一態様によれば、有効な命令長の短い命令実行後、有効な命令長の長い命令を実行する時、及び、有効な命令長の長い命令の実行後、有効な命令長の短い命令を実行する時には、両者の間に必要な

8

だけの無効命令を挿入することによりソフト的に両者の競合を解消できるのでハードウェアで両者の競合を検出する必要はない。ハードウェアが検出しなくてはならないのは、現サイクル以前に実行した命令長の短い命令と現サイクルに実行する命令長の短い命令との間の競合だけである。故に、本発明によれば1マシンサイクルで実行される命令長の短い命令の数を命令長の長い命令の中に指示される演算の数より小さくすることにより、平均的に1マシンサイクルに実行される演算数が高い割に、現サイクル以前に実行した、現サイクルに実行した命令との間のハードウェアによる競合検出を容易にできる。

【0030】本発明の他の一態様によれば、命令によって指示された演算結果は直ちに次命令に反映されず、一定数後の命令から反映されるので、命令実行後、その結果が反映されるまでに実行される命令は、書き込まれる前のレジスタの値を読むことができ、ソフトウェアが使うレジスタの数を実質的に多くし、ソフトウェア上の最適化により演算の並列度をあげることができる。

【0031】本発明の他の一態様によれば、競合検出のためのハードウェアが簡単になり、マシンサイクルの向上が図られ、処理速度を高めることができる。

【0032】本発明の他の一態様によれば、単一演算を指示する命令長の短い従来アーキテクチャの命令に、複数演算を指示する命令長の長い命令を追加して新アーキテクチャとできるので新アーキテクチャに従来アーキテクチャを含ませ、上位互換性を保つことができる。

【0033】

【実施例】次に本発明の好ましい一実施例について述べる。発明の本質と無関係な詳細は省略してある。

【0034】図1に全体ブロック図を示す。1200はメモリ、1300は命令キャッシュ、1303は命令制御ユニット、160は演算ユニット、150は命令長判定手段、109は競合検出手段である。命令制御ユニット1303は、インタフェース170を用いて、命令キャッシュより命令を読み出し、デコードし、インタフェース180を通じて演算ユニット160を制御する。演算ユニット160は複数の演算を並列に処理することができる。本計算機は単一演算を指示する4バイト長命令と、複数演算を指示する16バイト長命令を有し、命令キャッシュ1300には、16バイト長命令間、及び、16バイト長命令と4バイト長命令の間の競合は無いように、16バイト長命令と4バイト長命令が混在しておかれている。競合検出手段109は、4バイト長命令間のみ競合を検出する。命令制御ユニット1303は、命令長判定手段150を具備し、16バイト長命令実行時は競合検出手段109の出力を無視し、4バイト長命令実行時には、競合検出手段109の出力に応じて、セレクト110は並列して実行できる演算を選びデコードし、インタフェース180を通じて演算ユニット160を制御する。尚、ここでは、演算ユニットが2つの場合

を示しているが、2つ以上でも良い。

【0035】以下、レジスタ構成、命令フォーマットを説明し、さらにパイプライン、及び、動作タイミングを説明し、最後に図1の全体ブロック図の詳細を述べる。

【0036】図2にレジスタ構成を示す。FR0~FR31は64ビット長の浮動小数点レジスタ、R0~R31は32ビット長の整数レジスタである。簡単のため、浮動小数点データは全て倍精度で64ビット長とする。また、アドレスは32ビット毎に振られているものとする。

【0037】本実施例では、命令長の短い命令の命令長を1語長、命令の長い命令の命令長を4語長とする。

【0038】図3に、命令形式を示す。1語は32ビットである。基本命令、分岐命令、ロード・ストア命令は1語長命令、複合命令は4語長命令である。基本命令は全てレジスタ・レジスタ間演算である。本実施例では、命令長の長い命令の命令長を4語長としたが、実施例によっては、もっと長いことも短いこともあり得る。

【0039】本実施例では、簡単のため4語長命令は必ず4語長境界で区切られた4語に配置されると仮定するが、この仮定をはずすことは容易である。

【0040】まず基本命令について説明する。OPフィールドはオペコードの種類を、S1とS2フィールドは2つのソースレジスタの番号を、Tフィールドはターゲットレジスタの番号を、CCフィールドは、フラグの立て方を示すフィールドである。即ち、S1とS2で示されるレジスタが、OPで示される演算をほどこされ、Tで示されるレジスタに結果が書き込まれる。詳細を図4に示す。

【0041】次に分岐命令について説明する。dはディスプレイースメントを示す。分岐命令では、プログラムカウンタPCにdの値が加算される。

【0042】次にロード・ストア命令について説明する。Fフィールドは、ロード、又は、ストアするデータが浮動小数点データであるか、整数データであるかを示す。

【0043】SIZEフィールドは、図4に示すように、ロード、又は、ストアするデータの語長を示す。整数については、1ワードのみが定義され、浮動小数点については、2~16ワードが定義されているものとする。図4に示すように、FST命令では、FR(S1)がR(S2)番地に書き込まれる。SIZEが16ワードの時には、FR(S1)~FR(S1+7)が、R(S2)番地から始まる連続する16ワードに書き込まれるものとする。また、FLD命令では、R(S1)+R(S2)番地のデータを、FR(T)に書き込む。SIZEが16ワードの時には、R(S1)+R(S2)番地から始まる連続する16ワードが、FR(T)~FR(T+7)に書き込まれる。

【0044】次に、図3、図5を用いて、4語長の複合

命令について説明する。この命令では、I1, I2, IT, SIZE, Fフィールドで示される、ロード・ストア操作と、J1, J2, JT, Jフィールドで示される整数演算と、M1, M2, MTフィールドで示される第1浮動小数点演算と、A1, A2, AT, Aフィールドで示される第2浮動小数点演算と、N1, N2, NTフィールドで示される第3浮動小数点演算と、B1, B2, BT, Bフィールドで示される第4浮動小数点演算と、CC, d, Nフィールドで示されるフロー制御の計7つの演算が指示できる。各フィールドの詳細については、図5に示す。第1浮動小数点演算と、第3浮動小数点演算は乗算、第2浮動小数点演算と第4浮動小数点演算は加減算である。Nフィールドは、本命令に続き、挿入したい無効サイクルの数を示す。使用方法については後で述べる。

【0045】整数演算について説明する。Jフィールド≠1111のときは、図5に示すように通常の演算を行う。しかし、Jフィールド=1111のときは、データの格納されているメモリからキャッシュメモリへのプリフェッチを行う。すなわち、R(J1)+R(J2)をアドレスとしてキャッシュメモリをアクセスして、該当するデータが無ければメモリからキャッシュへデータを転送する。

【0046】また、1語長命令では、演算結果は次命令に直ちに反映されるが、4語長命令では、演算結果は3つ後の命令に初めて反映される。この仕様を好ましく利用したパイプライン構成と、プログラム例について以下述べる。

【0047】図6に示すようにパイプライン構成は、IF, D, E, F, Sの5段である。IFステージでは、命令の読み出し、Dステージでは、命令のデコード、Eステージではレジスタの読み出しと演算の一部、Fステージでは、演算、Sステージでは、演算の残りレジスタへの演算結果の書き込みが行われる。パイプライン構成は、整数演算と浮動小数演算で同じとする。

【0048】図7に1語長命令の本実施例による処理フローを示す。1マシンサイクルに2命令処理されるスーパスカラ方式である。命令1と2、命令3と4、命令5と6、命令7と8のそれぞれが、特に競合の無い限り、並列に処理される。このスーパスカラ方式については、特願昭63-283673号に詳細に記されている。

【0049】次に図8は、命令2の結果を命令3が使う場合の処理の様子を示したものである。命令3と4のEステージは、命令2のSステージが終了するまで引き伸ばされる。前命令の結果が次命令に反映するという命令仕様を満足するため、ハードウェアは上記競合を検出し、図8に示す処理を行わなくてはならない。

【0050】4語長命令の処理の様子を図9に示す。4語長命令は、1マシンサイクルに1命令ずつ処理される。命令1の演算結果は先に述べた仕様により、命令

2, 3には反映されず、命令4になって初めて反映される。命令1のレジスタ書き込みステージであるSステージは、命令4のレジスタ読み出しステージであるDステージの1つ前にちょうど終了しているので、図8を用いて説明した様なハードウェアによる競合制御は必要ない。本実施例では、演算ステージはE, F, Sの3段であるが、一般に、演算結果を書き込む前に行う次命令の数Nとパイプライン段数Mの間に $N \geq M - 1$ であればハードウェアによる競合制御は不要である。本実施例では $N = 2$, $M = 3$ のケースである。

【0051】また、1語長命令と次の有効な4語長命令の間には必ず無効な4語長命令を2つおくものとする。同様に有効な4語長命令と次の1語長命令の間には必ず無効な4語長命令を2つおくものとする。

【0052】次にこの4語長命令の好ましいプログラム例について図10, 図11, 図12を用いて述べる。次式の計算をする場合を考える。

【0053】

$$A(i) = A(i) + C \times B(i), \quad 1 \leq i \leq 24$$

但し、Cは定数、A(i), B(i)は、メモリ上に、図10のように配置されている64ビットの浮動小数点データである。

【0054】図11は、A(i)の計算をするのに、各サイクルに、どんな演算がされるかを説明する図である。横軸は時刻で、単位はマシンサイクルである。図に書かれた横長の箱は、処理されるデータの通る演算器のE, F, Sの3ステージを示す。(1)~(10)について説明する。演算は、インデックス1について4つずつ行われる。(1)~(10)はA(9)~A(12)を計算するための処理である。以下、各処理について説明する。定数CはFR31にあるものとする。

【0055】(1) A(1)~A(4)をFR4~FR7, B(7)~B(10)をFR0~FR3にロードする。

【0056】(2) $FR0 \times FR31$ をFR8に格納。

【0057】(3) $FR1 \times FR31$ をFR9に格納。

【0058】(4) $FR4 + FR8$ をFR12に格納。

【0059】(5) $FR5 + FR9$ をFR13に格納。

【0060】(6) $FR2 \times FR31$ をFR10に格納。

【0061】(7) $FR3 \times FR31$ をFR11に格納。

【0062】(8) $FR6 + FR10$ をFR14に格納。

【0063】(9) $FR7 + FR11$ をFR15に格納。

【0064】(10) FR12~FR15をA(9)~A(12)にストアする。

【0065】(1)~(10)の演算スケジューリングに関して、1つの演算に、図10を用いて説明した様に3サイクルかかることが考慮されている。A(9)~A

(12)の処理について説明したが、A(13)~A(16), A(17)~A(20)の処理も全く同様に行われる。各演算器は1サイクルピッチでパイプラインされており、A(13)~A(16), A(17)~A(20)の処理をA(9)~A(12)の処理に重ねて、図11のように処理可能である。

【0066】さて、図11に示した処理を実現する4語長命令列を示したのが図12である。A(1)~A(24)の演算は、図12に示す命令1~命令22の22命令で実現できる。使用するレジスタはFR0~FR15, FR31の17本である。命令1, 3, 5, 7, 9, 11, 13, 15でどのデータがロードされるか、又、命令12, 14, 16, 18, 20, 22でどのデータがストアされるかを、図10に示した。命令3はFR0~FR3に値を書き込むか、その結果が反映するのは命令6以降であるので、命令1でロードしたFR0~FR3の値を命令4で使用することが可能である。命令1の結果が、命令2にすぐに反映する従来の方式で図11と同じ処理を行おうとすると、命令3でFR0~FR3に書き込みを行うことはできず、FR16~FR19等、新たなレジスタが必要となる。ところが、使用できるレジスタの数には限りがあり、レジスタの数がネックになり、処理サイクル数が伸びてしまう。図12のプログラムは、現命令の結果が3命令後にしか反映しないという遅延書き込みを生かした結果、17本のレジスタを使うだけで、演算が可能となったのである。

【0067】本例で示した様に遅延書き込みは、レジスタを指定するオペコードのフィールドを増加させずに、実質的に使用できるレジスタの数を増やす効果がある。

【0068】図12上で、“X”印は何も演算するものがない為、空いてしまったフィールドであるが、特に、処理の開始時の命令1~6, 終了時の命令17~22に空フィールドが多い。しかし、これらの空フィールドは、コンパイラにより、次の一連の処理の開始処理と、現在の一連の処理の終了処理を重ねることにより、減らすことができる。また、全くすることのない、命令2, 4, 21は、命令1, 3, 20のNフィールドを“01”にすることにより、省略することができる。

【0069】本発明では、演算結果を書き込むレジスタの指定は、演算を指定する命令で行うので、命令4, 21が初めて省略できる。演算結果を書き込むレジスタの指定を、書き込みが行われるステージに発行される命令で指定する方式では、命令4, 21では命令1, 18で演算した結果の書き込み指定が必要であり、省略不可である。

【0070】また、図12のプログラムの前に1語長命令があった時に、図12の命令1の前に2つの4語長の無効命令を挿入しなくてはならないが、これは、Nフィールドが“01”である4語長の無効命令を1つ入れればよい。また、図12のプログラムの後に1語長命令が

13

来る場合、図12の命令22のNフィールドを“10”にすればよい。

【0071】このように命令を長くしてNフィールドを設け、有効に使用することにより、コードサイズを削減することが可能である。また、1語長命令では4語で4つの演算しか指示できないのに比べ、4語長命令では4語で、図5に示すように7つの演算が指示できる。

【0072】次に、プログラム作成方法について述べる。プログラムはFORTRAN、Cなどの高級言語で記述し、コンパイラにより命令列に変換する。

【0073】図26に本発明に関するコンパイラの処理フローを示す。高級言語によるプログラムは、字句解析部、構文解析部、意味解析部を経て、中間コードに変換される。中間コードは、最適化部によって最適化され、コード生成部により図3に示した命令列に変換される。ここで、最適化部とコード生成部を合わせて合成部という。この合成部に本願発明に関する特徴がある。すなわち、この合成部では、中間コードを見て並列してできる演算ができるだけ多くなるように命令列を生成する並列化部を有する。この並列化部では、並列してできる演算数が多いときには4語長命令を用い、並列してできる演算数の少ないときには1語長命令を用いる。ここで、4語長命令を用いるか1語長命令を用いるかの判断基準は、並列して実行できる演算数により決まるが、この演算数はシステムによって異なるので、プログラムを作成するときにパラメータとしてコンパイラに指定できるようになっている。上述のようにすることにより、コードサイズが小さくなり、主メモリ、及び、キャッシュメモリの使用効率が上がり、処理速度が高められる。

【0074】次に、合成部の特徴としてレジスタの割当てがある。4語長命令はその演算結果が3つ後の命令に初めて反映されるので、特別な配慮が必要になる。例えば、図12において命令1の結果は命令4に初めて反映されるので、命令2、3では命令1の結果を使用せずにできる演算をできるだけたくさん割り付けておく。この時、演算できる命令が1つも無いときは、無効命令生成部によって無効命令を挿入しておく。また、合成部では、1語長命令と次の有効な4語長命令との間には必ず無効な4語長命令を2つ挿入しておく。逆に、有効な4語長命令と次の1語長命令との間にも必ず無効な4語長命令を2つ挿入しておく。ここで、先に述べたように無効命令はNフィールドを用いることによって省略できる。すなわち、本実施例のコンパイラは、長い命令間の競合を検出し、Nフィールドを用いて、命令実行後に挿入すべき無効サイクルの数を指定できるので、ハードウェアで長い命令間の競合を検出したり、処理をする必要が無い。

【0075】次に、これまでに説明した命令の処理を行うハードウェアの一実施例について説明する。図13は図1を詳細化した全体構成である。1300は命令キャ

14

ッシュ、1301は命令キャッシュコントローラ、1302は命令処理フローを制御する分岐ユニット、1303は、命令をデコードする命令制御ユニット、1304は整数演算ユニット、1307は浮動小数点演算ユニット、1306はデータキャッシュ、1305はデータキャッシュコントローラ、1308はメモリインタフェースユニットである。

【0076】命令制御ユニット1303は、命令キャッシュ1300より実行すべき命令をバス1310を通して受け取り、命令をデコードし、整数演算ユニット制御信号1318を整数演算ユニット1304に、浮動小数点演算ユニット制御信号1314を浮動小数点演算ユニット1307に、分岐ユニット制御信号1312を分岐ユニット1302に送出する。さらにプログラムカウンタ3500の制御のためにモード信号110も、分岐ユニット1302に送出する。また、整数演算ユニット1304より、フラグ1317を、浮動小数点演算ユニットよりフラグ1315を受け取る。

【0077】整数演算ユニット1304は、オペランドアドレス1319をデータキャッシュ1306と、データキャッシュコントローラ1305に送出する。データキャッシュより読み出されたデータは、データバス1320を通して、整数演算ユニット1304、又は、浮動小数点演算ユニット1307に送出される。データキャッシュの中に所望のデータが無い時には、データキャッシュコントローラ1305が、メモリインタフェースユニット1308にインタフェース信号1321を通して起動をかけ、主メモリよりデータを読み出す。この間の待合わせ制御を、信号1316を通して、命令制御ユニット1303と行う。

【0078】分岐ユニットは、次に読み出すべき命令のアドレス1309を、命令キャッシュ1300と、命令キャッシュコントローラ1301に送出する。所望の命令が命令キャッシュ1300の中に無い時には、命令キャッシュコントローラ1301は、メモリインタフェースユニット1308にインタフェース信号1313を通して起動をかけ、主メモリより、命令を読み出す。この間の待合わせ制御を信号1311を通して、命令制御ユニット1303と行う。

【0079】整数演算ユニット1304の詳細を示したのが、図14である。1400はデコーダ、1401は第1ALU、1402は第2ALU、1403は整数レジスタである。第1ALUにはソースバス1406、1407を通して、整数レジスタファイル1403からデータが送られ、演算結果は、ターゲットバス1322を通して、整数レジスタファイル1403に返される。また、第2ALUには、ソースバス1408、1409を通して、整数レジスタファイル1403からデータが送られ、演算結果は、ターゲットバス1319を通して整数レジスタファイル1403に返される。1317-1

は、第1ALUより出力されるフラグ、1317-2は第2ALUより出力されるフラグである。バス1319、1322は、ロードストア及びプリフェッチ時のアドレスとしてデータキャッシュ1306に送出される。

【0080】図13の浮動小数点演算ユニット1307の詳細を示したのが、図15である。1501はデコーダ、1502は浮動小数点レジスタファイル、1503は第1乗算器、1504は第2乗算器、1505は第1加算器、1506は第2加算器である。第1乗算器1503には、ソースバス1517、1518を通して、第2乗算器1504には、ソースバス1515、1516を通して、第1加算器1505には、ソースバス1513、1514を通して、第2加算器1506には、ソースバス1511、1512を通して、浮動小数点レジスタファイル1502よりデータが送られ、演算結果はそれぞれ、ターゲットバス1507、1508、1509、1510を通して浮動小数点レジスタファイルに書き込まれる。

【0081】1315-1は、第1乗算器1503のフラグ、1315-2は第2乗算器1504のフラグ、1315-3は第1加算器1505のフラグ、1315-4は第2加算器1506のフラグである。

【0082】図15の浮動小数点レジスタファイル1502の詳細を示したのが、図16である。1600~1608は浮動小数点レジスタ、1314-1~1314-9はそれぞれ浮動小数点レジスタ1600~1608の制御信号である。1610はロードアライナ、1609はストアアライナである。1611~1618は、浮動小数点レジスタ1600~1608を、メモリと結ぶバスである。バス1611は、FR0、16、24に、バス1612は、FR1、9、17、25に接続されている。バス1613、1614、1615、1616、1617も同様で、バス1618にはFR7、15、23、31が接続されている。ロード命令実行時には、バス1320を通して、送られてきたデータを、ロードアライナ1610により、1611~1618の内、所望のバスに乗せかえ、所望のレジスタに書き込む。またストア命令実行時には、1611~1618にレジスタよりデータが読み出され、ストアアライナ1609により、バス1320の所望の位置にデータが出力される。

【0083】図16の浮動小数点レジスタ1600の第1の実施例を記したのが、図17である。レジスタ1600について示したが、1601~1608も同様である。図17に示すようにレジスタ1600は64ビットのレジスタの集まりである。1700~1763はそれぞれ1ビットのレジスタである。1511-00~1518-00はレジスタ1700の読み出しバス、1507-00~1510-00はレジスタ1700の書き込みバス、1611-00はレジスタ1700の読み書きバスである。レジスタ1763のバス構成も同様であ

る。

【0084】図28は、図13のデータキャッシュの詳細を図示したものである。2801はデータを保持するデータアレイ、2800はロードストア演算用のアドレスアレイ、2802はプリフェッチ用のアドレスアレイである。アドレスアレイ2800と2802は同じ内容のデータを保持している。1語長命令のロードストア命令実行時には、バス1319または1322を用いてアドレスアレイ2800とデータアレイ2801がアクセスされる。4語長命令実行時には、バス1322を用いてアドレスアレイ2800とデータアレイ2801がアクセスされる。また、バス1319を用いてプリフェッチのためにアドレスアレイ2802がアクセスされる。

【0085】図29は、ロードストア演算でキャッシュミスを生じたときのパイプラインを示す図である。パイプラインは、データがメモリからキャッシュメモリへ転送される間ロックされる。図29で示されるのがロックされる期間である。

【0086】一方、プリフェッチ演算のときには、アドレスアレイ2802がヒットすれば何も行わない。ミスすれば、そのアドレスを含むブロックがメモリよりデータアレイ2801にバスを通して転送される。但し、この時はパイプラインはロックされない。コンパイラにより、ミスする可能性のあるロードストア演算の前にプリフェッチを設定しておけば、メモリからキャッシュメモリへの転送を他の演算と並列に行うことができる。そのために、図29で示したパイプラインロックによる性能低下を避けることができる。

【0087】図17のレジスタ1700の回路構成例を記したのが、図18である。1816と1817はインバータ、1802~1815はクロックドインバータである。

【0088】1314-1-1~8がhighになると、それぞれバス1511-00~1518-00にレジスタの値が出力される。また、1314-1-10~14がhighになると、バス1510-00~1507-00の値がレジスタに書き込まれる。また、1314-1-9がhighになるとレジスタの値がバス1611-00に出力され、1314-1-10がhighになると、バス1611-00がレジスタに書き込まれる。信号1800は予備の読み出しポート、1801は予備の書き込みポートである。1800と1801の用途については後に説明する。

【0089】図19は、図16の浮動小数点レジスタ1600の第2の実施例を示したものである。図19の実施例は、図17の実施例と比較して1900~1963の第1シャドウレジスタ、2000~2063の第2シャドウレジスタが付加されている点が異なっている。第1シャドウレジスタ1900は信号1800を通して、レジスタ1700の値を読み取ることができる。また、

17

信号1964を通して第2シャドウレジスタ2000に、第1シャドウレジスタ1900の値を送出する。第2シャドウレジスタ2000は、自分の値を信号1801を通してレジスタ1700に送出する。即、レジスタ1700~1763、第1シャドウレジスタ1900~1963、第2シャドウレジスタ2000~2063はリング状のシフトレジスタを構成している。更に第1シャドウレジスタ1900~1963、第2シャドウレジスタ2000~2063は、レジスタ1700~1763と同様に、バス1611-00~1611-63を通して、読み書きができる。

【0090】1314-1-15は、第1シャドウレジスタ1900~1963の制御信号、1314-1-16は、第2シャドウレジスタ2000~2063の制御信号である。

【0091】シャドウレジスタの目的は、4語長命令実行時の割込みからの復帰を可能にすることである。図20~図22を用いてその動作を説明する。W'ステージは、レジスタから、第1シャドウレジスタFRS1に書き込むステージ、W''は第1シャドウレジスタFRS1から第2シャドウレジスタFRS2に書き込むステージである。

【0092】図20は割込みのない通常時の4語長命令の動作である。FR、FRS1、FRS2のタイムチャート上の数字は、どの命令の演算結果が各レジスタに入っているかを示す。図20の通り、通常時は、FRからFRS1へ、FRS1からFRS2へと1サイクルビッチで演算結果がシフトされる。

【0093】図21は、命令3と命令4の間に割込みが入った時の動作を示す図である。命令4、5、6、7は無効化される。各レジスタは割込み発生後値の更新を止め、FRは命令3の結果を、FRS1は命令2の結果を、FRS2は命令1の結果を保持する。また、プログラムカウンタには割込みベクタがセットされる。割込みベクタから始まる割込み処理プログラムで、FR、FRS1、FRS2の値をメモリ上に退避する図22は、割込み処理からの復帰時の動作を説明する図である。まず、割込み処理プログラムの最後に、図22に示すように命令1の結果をFRに、命令2の結果をFRS2に、命令3の結果をFRS1に復帰する。こうすることにより、命令4のレジスタ読み出しステージのEステージで、命令1の結果を見ることができる。命令4のEステージ終了後、FRの値をFRS1に、FRS1の値をFRS2に、FRS2の値をFRへコピーする。この結果、命令5のEステージでは命令2の結果を見ることができる。命令5のレジスタ読み出しステージのEステージ終了後も、同じ動作をさせることにより、命令6のレジスタ読み出しステージのEステージで、命令3の結果を見ることができる。以後の処理は通常通りで、1命令実行毎にFRの値をFRS1へ、FRS1の値をFRS

18

2へコピーし、FRS2の値は捨てる。

【0094】以上、説明した様に、シャドウレジスタを設けることにより、遅延書き込み命令実行時でも割込みを受け付け復帰することができる。シャドウレジスタの無い場合は、図21で命令3の結果しか退避できず、図22の割込み復帰時の命令4で、命令1の結果を見ることができない。これは、命令2、3が命令1と同じレジスタに値を書き込む場合があるからである。例えば図12のプログラムでも、命令1と同じレジスタに命令3で書き込む。

【0095】次に、シャドウレジスタ付加によるハードウェアの増加量について述べる。レジスタの大きさは、ポート数にほぼ比例する。図17と図19を比べれば分かるように、シャドウレジスタのポート数は3と、レジスタのポート数13に比べてずっと小さいので、シャドウレジスタ付加によるハードウェアの増加量は小さい。

【0096】図41は、図13の命令制御ユニット1303の実施例である。150は命令語長判定手段、101は第1命令レジスタ、102は第2命令レジスタ、103は第3命令レジスタ、104は第4命令レジスタである。4100はモードレジスタである。100はモード制御回路、105はレジスタ読み出し制御回路、106はレジスタ書き込み制御回路、107はファンクション制御回路、108はパイプライン制御回路、109は競合検出回路である。

【0097】4語長命令は、必ず4語境界をまたがない様に配置されているものとする。また、1語長命令は、2語境界に囲まれた2語が同時に実行されるものとする。本実施例では命令語長判定は、図3で説明した様にオペコードの中の最も左のビット、即、図41の信号1310-1-1(C000)そのものを見ることにより行われる。

【0098】モード制御回路100の詳細を示したのが図27である。2700は制御回路、2701はNフィールド1310-4-1を保持するレジスタ、2702はディクリメンタ、2703はコンパレータである。コンパレータの出力信号VALID(2704)は、制御回路2700に送り出される。レジスタ2701にセットされたNフィールドの値は、ディクリメンタ2702により1サイクルごとに1減算され、00になったときに信号VALID(2704)がアサートされる。信号VALIDはネゲート時に無効サイクルの挿入を指示し、アサート時に命令の実行を指示する信号である。

【0099】制御回路2700は、競合検出回路出力116(BUB)と、命令アドレスの下位から2ビット目1309-1(CA30)と、オペコード中の4語長命令かどうかを示すビット1310-1-1(C000)と信号2704(VALID)を見て5つのモードの内、どれであるかを判定し、図23に示すように第1~4命令レジスタへのオペコードのセット、プログラムカウンタの

インクリメントを信号110により行う。また、現サイクルがどのモードであるかを示す信号110は、モードレジスタ4100にラッチされ、その出力信号130はレジスタ読み出し制御回路105、レジスタ書き込み制御回路106、ファンクション制御回路107、パイプライン制御回路108、競合検出回路109に送出される。図23で、C0~3は、4語境界内にある4語で、若いアドレスよりC0、C1、C2、C3と命令が並んでいるものとする。C0の最左ビットC000により、図3に示すように、1語長命令か4語長命令かが判定できる。1語長命令モード1は、4語境界内の左の2つの命令(C0、C1)を実行するモードで、第1命令レジスタにC0が、第2命令レジスタにC1がセットされプログラムカウンタPCは+2される。また、1語長命令モード2は、4語境界内の右の2つの命令(C2、C3)を実行するモードで、第1命令レジスタにC2、第2命令レジスタにC3がセットされ、プログラムカウンタPCは+2される。即ち、1語長命令実行時には、第1命令レジスタと、第2命令レジスタのみ用い、第3命令レジスタと第4命令レジスタは用いない。4語長命令モードは、4語長命令(C0、C1、C2、C3)実行するモードで、第1~4命令レジスタにC0~C3がセットされ、プログラムカウンタPCは+4される。競合モードとは、競合検出回路109が競合を検出した場合で、第1~4命令レジスタ及び、モードレジスタ4100は前サイクルの値を保持する。また、プログラムカウンタPCの更新は行わない。無効命令モードは、現サイクル以前に実行した4語長命令のNフィールドで、現サイクルにハードウェアで無効命令(NOP)を挿入することを指示されている場合で、命令レジスタには無効命令がセットされ、プログラムカウンタPCは更新されない。これにより、無効サイクルが1サイクル挿入されることになる。

【0100】1語長命令実行時にはC0、又は、C2の実行の為に、第1ALU1401(図14)、第1乗算器1503(図15)、第1加算器1505(図15)を用いる。一方、C1、又は、C3の実行の為に、第2ALU1402(図14)、第2乗算器1504(図15)、第2加算器1506(図15)を用いる。また、4語長命令実行時には、ロードストア演算のアドレス計算を第1ALU1401(図14)で、整数演算を第2ALU1402(図14)で、第1浮動小数点演算を第1乗算器1503(図15)で、第2浮動小数点演算を第1加算器1505(図15)で、第3浮動小数点演算を第2乗算器1504(図15)で、第4浮動小数点演算を第2加算器1506(図15)で行う。

【0101】図41のレジスタ読み出し制御回路105、レジスタ書き込み回路106、ファンクション制御回路107は、モード制御回路出力のモード指定信号110と第1~4命令レジスタの値により、上述の演算器割当て規則に従い、整数演算ユニット1304(図13)

の制御信号1318を生成する。レジスタ読み出し制御回路について、更に詳細に説明したのが図24である。6つの演算器のそれぞれの2つの入力に入れるレジスタの指定を、オペコードのどのフィールドで行うかを示している。フィールドの略号については、図3の複合命令の欄に示す。4語長命令のJ1とA1は、1語長命令のS2と、4語長命令のJ2とA2は、1語長命令のS2と同じ位置であることを利用し、図24上のフィールド指定には、1語長命令時も、J1、J2、A1、A2を用いて述べている。これは、C0とC1の区別の為である。

【0102】次に図41の競合検出回路109について述べる。図7~図9を用いて説明したように、本実施例では4語長命令間の競合を検出する必要はない。また、本実施例では、1語長命令実行用の全ての演算器を2重化している為、同時に実行する2つの1語長命令間の演算器による競合はあり得ない。簡単のためにレジスタ競合も無いものとする。本実施例をレジスタ競合がある場合に拡張することは、例えば、特願昭63-283673号のように容易である。前に述べたように、1語長命令と次の有効な4語長命令の間に、必ず無効な4語長命令が2つ入り、同様に、有効な4語長命令と次の1語長命令との間には必ず無効な4語長命令が2つ入っているため、4語長命令と1語長命令間の競合も有り得ない。故に、競合検出回路109は、現サイクルの1語長命令と、それ以前に実行された1語長命令間の競合のみを検出すればよい。モード制御回路100の制御により、1語長命令は、第1命令レジスタ101と第2命令レジスタ102にのみセットされるので、競合検出回路109は、第1命令レジスタ101と第2命令レジスタ102のみを見ればよく、第3命令レジスタ103、第4命令レジスタ104は見る必要はない。

【0103】図25は、競合検出回路109の実施例のブロック図である。2501~2504はレジスタ、2505はマスク回路、2506~2521はコンパレータである。図7において、命令7、8を現命令とし、命令3~6との競合検出を考える。命令1、2のSステージの次に命令7、8のEステージが来るので、命令1、2と命令7、8の間の競合はない。図25のレジスタ2501には、命令5が書き込むレジスタの番号が、レジスタ2503には、命令6が書き込むレジスタの番号が、レジスタ2502には、命令3が書き込むレジスタの番号が、レジスタ2504には命令4が書き込むレジスタの番号が記憶されている。上記4つのレジスタと、命令7及び命令8が読み出す4つのレジスタの番号を、2506~2521の16個のコンパレータで比較し、結果をマスク回路2505に送出する。マスク回路2505は、モード制御回路100の出力130や、パイプライン制御回路108の出力115を見て、コンパレータのヒット信号が有効であるかどうかを判定し、有効であ

21

ば競合を示す信号116をアサートする。即、コンパレータの出力がレジスタの一致を示していてもその命令が無効化される場合は、信号116をネゲートする。このマスク回路2505により、モード信号130が4語長モードであることを示している時には、信号116をネゲートする。

【0104】次にパイプライン制御回路108について説明する。パイプライン制御回路は、モード信号130、図13の整数演算ユニット1304からのフラグ信号1317、図13の浮動小数点演算ユニット1307からのフラグ信号1315、図13のデータキャッシュコントローラとのインタフェース1316、図13の命令キャッシュコントローラとのインタフェース1311を用いて、図13の分岐ユニット1302制御信号1312を送出し、分岐ユニットを制御する。即ち、有効な分岐命令が来た時には、分岐を行い、それ以外の時には、モード信号110を用いて、図23のように分岐ユニット内にあるプログラムカウンタを制御する。またパイプライン制御回路108は、信号115を、レジスタ読み出し制御回路105、パイプライン書き込み制御回路106、ファンクション制御回路107、競合検出回路109を送出し、パイプラインの状態を制御する。即ち、命令キャッシュ、あるいはデータキャッシュのアクセスに際してミスを生じた時に図29に示すようにパイプラインをロックする。

【0105】次に上記実施例の第1の変形例について述べる。上記実施例において、4語長命令は演算結果を3つ後の命令に初めて反映することにより、4語長命令間の競合検出部が不要になった。同様の効果を達成するために、4語長命令でも演算結果を次の命令に反映するが、4語長命令間での競合をコンパイラで避けるようにすることもできる。具体的には、ある4語長命令が書き込むレジスタを次の4語長命令と次の次の4語長命令とで読まないようにするのである。このようにすることによって、第1の実施例のレジスタ数を実質的に増やす効果は失われるが、図19で示したシャドウレジスタは不要になる。

【0106】さらに、上記実施例の第2の変形例について述べる。図3で示した実施例では、命令の中に1語長命令か4語長命令かを示すビットを持っていたが、計算機の中に1語長命令か4語超命令かを示すフラグを持ち、このフラグを命令で制御することも可能である。これによって、フラグを制御する命令が必要になるが、1度フラグを切り換えれば毎回命令の中で語長を示す必要が無くなるという利点がある。

【0107】図30～図32を用いて、上記実施例の第3の変形例について述べる。

【0108】本変形例では、図30の様に浮動小数点レジスタを32本から128本に拡張している。図31に命令フォーマット、図32に命令の説明を示す。FR0

22

～31は基本命令、複合命令の両者で使用可であるが、FR32～FR127は、複合命令でのみ使用するレジスタである。レジスタを指定するI1, IT, MT, A1, AT, N1, NT, B1, BTの各フィールドは図31に示すように、各7ビットと増加する。本変形例では、ソースレジスタの片方をターゲットレジスタと一致させることにより、全体を4語に収めているが、そうしなくては、全体の語長をさらに長くすることも可能である。

【0109】本変形例では、基本命令に複合命令を追加し、複合命令で基本命令が使えるレジスタの数よりも、多くのレジスタを使えるようにすることにより、全体として、使用可能なレジスタ数を増やせるという効果がある。本変形例ではFR0～31は基本命令と複合命令の両方でアクセス可であるが代案として、基本命令用の32本と、複合命令用の128本のレジスタを独立のものとする事も可能である。

【0110】さらに本変形例では、図32に示すように、メモリからキャッシュへのプリフェッチの際にプリフェッチする語数をJTフィールドで指定可となっている。これにより一度に複数のブロック転送を行うよう命令で指示でき、効率が上がるという利点がある。

【0111】図33、図34を用いて上記実施例の第4の変形例について述べる。本変形例では、データのプリフェッチをJ1, J2, JTといった整数演算フィールドで行わずに、1ビットのPフィールドで行っている点異なる。P=1の時には、ロード・ストア演算に用いられたアドレスを含むブロックの次のブロックがプリフェッチされる。こうすることにより、プリフェッチに要するフィールドが節約でき、ロード・ストア演算、整数演算、プリフェッチの3動作が並列指定可となるという利点が生ずる。

【0112】図35はもう1つの実施例について説明する全体ブロック図である。

【0113】3500はプログラムカウンタ、3501は命令を格納するメモリ手段、3502はマスク・スイッチ回路、3503～3506はM個のnバイト長の命令レジスタ、3507はデコーダ、3508, 3509はL個(L≥1)の演算ユニット、150は命令長判定手段、109は競合検出回路、100はモード制御回路、4100はモードレジスタである。

【0114】プログラムカウンタ3500は命令アドレス3513を、命令を格納するメモリ手段3501に送出する。3501の中には、nバイト長の命令とn×Mバイト長(M>1)の命令が混在しており、命令アドレス3513で指定された命令を含む複数の命令をマスク・スイッチ回路3502に送出する。マスク・スイッチ回路3502はnバイト命令であればM個の命令レジスタの内N個(1≤N<M)の命令レジスタ3503～3504の内の少なくとも1つにセットし、n×Mバイト

命令であれば命令レジスタ3503~3506にセットする。デコーダ3507は命令レジスタ3503~3506よりの命令3519~3522をデコードし、L個の演算ユニットを制御信号3523、3524を用いて制御する。命令長判定手段150は、少なくとも命令3514の一部を見て、命令長を示す信号3526をモード制御回路100に送出する。競合検出回路109は、命令レジスタ3503~3504を見て、nバイト命令間の競合の有無を知らせる信号116をモード制御回路100に送出する。モード制御回路は、命令長、競合の有無、プログラムカウンタの値によりモードを判定し、制御信号110により、プログラムカウンタ、マスク・スイッチ回路、デコーダを制御する。

【0115】本実施例と、図1~図29及び図41の実施例との対応を説明する。図1~図29の実施例は、 $n=4$ 、 $M=4$ 、 $N=2$ 、 $L=2$ の場合である。また演算ユニットは整数演算ユニットと浮動小数点演算ユニットであった。また、図35のマスク・スイッチ回路は、図41の第1~4命令レジスタ101~104にセットする命令を生成しているセレクタや、NOPによるマスク等に対応する。また図35の競合検出回路109は、図41の競合検出回路109に対応する。図35の命令長判定手段150は図41の命令長判定手段150に対応する。図35のモード制御回路100は、図41のモード制御回路100に対応する。

【0116】図3、図5、図10、図11、図12で示した実施例では、図10の様にメモリ上でのデータ配置が制約されるという欠点があった。これを解決したのが図36~図40に示す実施例である。本実施例の複合命令では、図36、図37に示すように、1命令でロード・ストア等のメモリ演算を2個実行できる。ハードウェアとしてはキャッシュを2ポート化するか、あるいは、1マシンサイクルに2度アクセス可能な構成にすればよい。図39、図40に図11、図12に示したものと同一問題を解くプログラムを示す。

【0117】これまでに説明した実施例では、命令長の短い命令と長い命令を混在した計算機において、長い命令の演算結果を、以後の任意の命令から反映する。あるいは、長い命令は、次命令以降に続く無効命令の数を任意に指定できる。あるいは、長い命令は、メモリまたはキャッシュメモリからレジスタへデータを転送する第1のフィールドと、メモリからキャッシュメモリへデータを転送する第2のフィールドを設けるといった工夫を行ったが、これらの工夫は、長い命令のみを有するVLIW型計算機に対しても有効である。

【0118】

【発明の効果】以上説明したように、本実施例では、4語長命令を用いて、4語で7つの演算を指定できるが、競合検出は、 $4 \times 4 = 16$ 個のコンパレータで行うことができる。4語長命令間の競合検出をハードウェアで行

おうとすると、前サイクルの分岐演算を除く、6個の演算の書き込みレジスタと前々サイクルの同じく6個の書き込みレジスタと、現サイクルの12個の読み出しレジスタとの間の競合検出が必要で、 $(6+6) \times 12 = 144$ 個のコンパレータが必要となる。本実施例では、これに比して16/144のハードウェアで済むという利点がある。

【0119】本実施例では、4語長命令で指示される演算数7に対して、1マシンサイクルで処理される命令長の短い命令が2である。これにより、2演算分の競合検出回路で最大7演算の並列処理が得られるという効果がある。

【0120】本発明によれば、並列に実行する演算の数を増やし、性能を高めることができる。

【0121】本発明によれば、コードサイズを小さくすることができる。これによりコードキャッシュのヒット率が高まり、性能を高めることができる。

【0122】本発明によれば、並列に実行する演算内のハードウェアによる競合検出を容易にすることができる。これにより、マシンサイクルを高めること、ハード物量を減らし、コストを下げるができる。特に、長い命令の中で指定する演算数が大の時、この効果は著しい。

【0123】本発明によれば、現サイクル以前に実行した命令と、現サイクルに実行する命令との間のハードウェアによる競合検出、待合わせを容易にすることができる。これにより、マシンサイクルを高めること、ハード物量を減らし、コストを下げるができる。

【0124】本発明によれば、ソフトウェアが用いられるレジスタの数を実質的に多くし、ソフトウェア上の最適化により演算の並列度をあげ、性能を高めることができる。

【0125】本発明によれば、従来アーキテクチャとの上位互換性を保つことができる。

【図面の簡単な説明】

【図1】命令制御ユニットの全体図である。

【図2】レジスタ構成を示す図である。

【図3】命令形式を説明する図である。

【図4】1語長命令の動作を説明する図である。

【図5】4語長命令の動作を説明する図である。

【図6】パイプラインステージを説明する図である。

【図7】競合無の時の1語長命令処理のパイプラインを示す図である。

【図8】競合有の時の1語長命令処理のパイプラインを示す図である。

【図9】4語長命令のパイプラインを示す図である。

【図10】データのメモリ上での配置を示す図である。

【図11】4語長命令を用いた時の演算の様子を説明する図である。

【図12】4語長命令を用いたプログラムを説明する図

である。

【図13】全体ブロック図である。

【図14】整数演算ユニットのブロック図である。

【図15】浮動小数点演算ユニットのブロック図である。

【図16】浮動小数点レジスタファイルのブロック図である。

【図17】浮動小数点レジスタのブロック図である。

【図18】浮動小数点レジスタ1ビット分の回路図である。

【図19】浮動小数点レジスタのブロック図である。

【図20】シャドウレジスタの動作を説明する図である。

【図21】シャドウレジスタの動作を説明する図である。

【図22】シャドウレジスタの動作を説明する図である。

【図23】モード制御回路の動作を説明する図である。

【図24】レジスタ読み出し制御回路の動作を説明する図である。

【図25】競合検出回路のブロック図である。

【図26】コンパイラの処理フローである。

【図27】モード制御回路の詳細を示した図である。

【図28】データキャッシュの詳細を示したものである。

【図29】ロードストア演算でキャッシュミスを生じたときのパイプラインを示す図である。

【図30】その他の実施例を示す図である。

【図31】その他の実施例を示す図である。

【図32】その他の実施例を示す図である。

【図33】その他の実施例を示す図である。

【図34】その他の実施例を示す図である。

【図35】その他の実施例を示す図である。

【図36】その他の実施例を示す図である。

【図37】その他の実施例を示す図である。

【図38】その他の実施例を示す図である。

【図39】その他の実施例を示す図である。

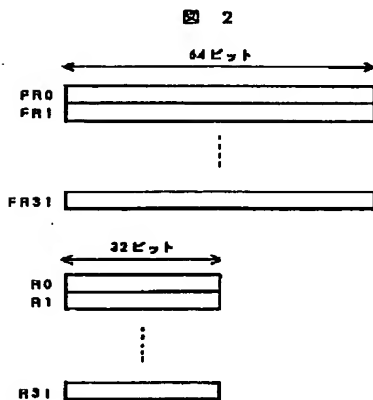
【図40】その他の実施例を示す図である。

【図41】図1を詳細化した図である。

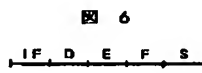
【符号の説明】

100…モード制御部、101…第1命令レジスタ、102…第2命令レジスタ、103…第3命令レジスタ、104…第4命令レジスタ、105…レジスタ読み出し制御、106…レジスタ書き込み制御、107…ファンクション制御、108…パイプライン制御、109…競合検出部。

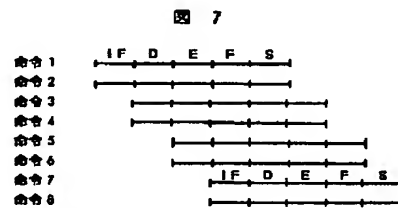
【図2】



【図6】

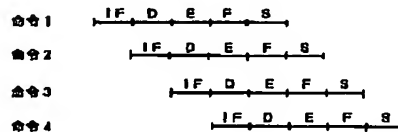


【図7】



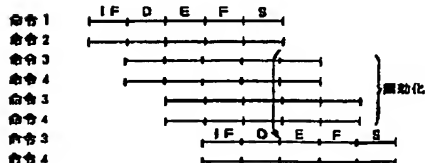
【図9】

図9



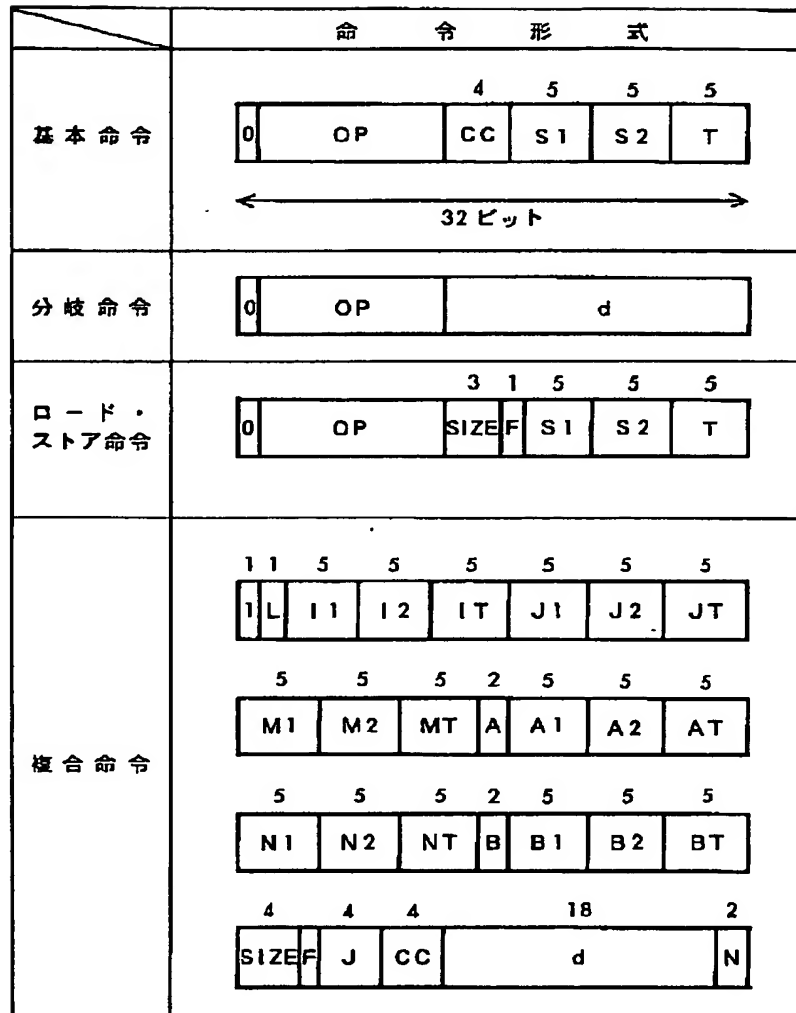
【図8】

図8



【図3】

図 3



【図29】

図 29



【図4】

図 4

	ニモニック	動 作	補 足															
基 本 命 令	ADD S1,S2,T	$R(S1)+R(S2) \rightarrow R(T)$	CCフィールド <table><tr><th>コード</th><th>意 味</th></tr><tr><td>0000</td><td>常に不成立</td></tr><tr><td>0001</td><td>常に成立</td></tr><tr><td>0010</td><td>=</td></tr><tr><td>0011</td><td>結果が正</td></tr><tr><td>0100</td><td>結果が負</td></tr><tr><td>⋮</td><td>⋮</td></tr></table>		コード	意 味	0000	常に不成立	0001	常に成立	0010	=	0011	結果が正	0100	結果が負	⋮	⋮
	コード	意 味																
	0000	常に不成立																
	0001	常に成立																
	0010	=																
	0011	結果が正																
	0100	結果が負																
	⋮	⋮																
	SUB S1,S2,T	$R(S1)-R(S2) \rightarrow R(T)$																
AND S1,S2,T	$R(S1) \cup R(S2) \rightarrow R(T)$																	
OR S1,S2,T	$R(S1) \cap R(S2) \rightarrow R(T)$																	
EOR S1,S2,T	$R(S1) (+) R(S2) \rightarrow R(T)$																	
NOT S1,T	$R(S1) \rightarrow R(T)$																	
FADD S1,S2,T	$FR(S1)+FR(S2) \rightarrow FR(T)$																	
FSUB S1,S2,T	$FR(S1)-FR(S2) \rightarrow FR(T)$																	
FMUL S1,S2,T	$FR(S1) \times FR(S2) \rightarrow FR(T)$																	
NOP	何もしない。																	
分 岐	BRA d	$PC+d \rightarrow PC$																
ロ ー ド ス ト ア 命 令	ST S1,S2	R(S1)をR(S2)でさされるメモリに書く	SIZEフィールド <table><tr><th>コード</th><th>意 味</th></tr><tr><td>000</td><td>1ワード</td></tr><tr><td>001</td><td>2ワード</td></tr><tr><td>010</td><td>4ワード</td></tr><tr><td>011</td><td>8ワード</td></tr><tr><td>100</td><td>16ワード</td></tr></table>		コード	意 味	000	1ワード	001	2ワード	010	4ワード	011	8ワード	100	16ワード		
	コード	意 味																
	000	1ワード																
	001	2ワード																
010	4ワード																	
011	8ワード																	
100	16ワード																	
LD S1,S2,T	$R(S1)+R(S2)$ でさされるメモリ上のデータをR(T)に書く																	
FST S1,S2	FR(S1)をR(S2)でさされるメモリに書く																	
FLD S1,S2,T	$R(S1)+R(S2)$ でさされるメモリ上のデータをFR(T)に書く																	

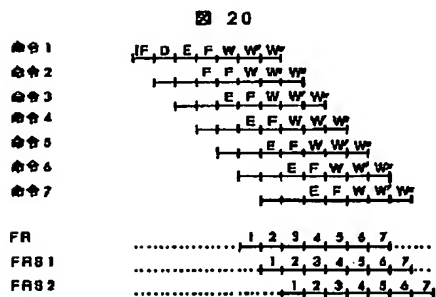
注) \cap …ビット毎のAND \cup …ビット毎のOR $(+)$ …ビット毎の排他的OR
 PC…プログラムカウンタ R(x)…レジスタ番号xの汎用レジスタ
 FR(x)…レジスタ番号xの浮動小数点レジスタ

【図5】

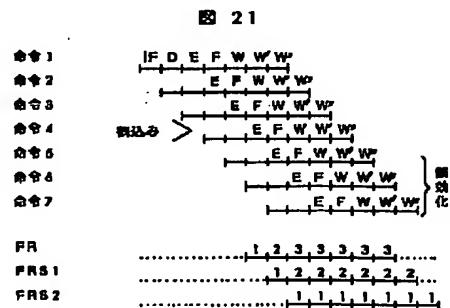
図 5

	動 作	補 足								
ロード・ストア演算	I1, I2, SIZ, F, ITフィールドを基本命令のS1, S2, SIZ, F, ITフィールドをみなして、ロード、ストアを実行	L=1…ロード、L=0…ストア F=1…FR、F=0…R								
整 数 演 算	J≠“1111”の時 R(J1)とR(J2)にJフィールドで示される演算を施して、R(JT)に格納	Jフィールド 0000 ADD, 0001 SUB 0010 AND,								
	J=“1111”の時 R(J1)+R(J2)で与えられるデータが、キャッシュに無ければ、メモリからキャッシュに転送									
第1浮動 小数点演算	FR(M1)×FR(M2)→FR(MT)									
第2浮動 小数点演算	FR(A1)とFR(A2)にAフィールドで示される演算を施してFR(AT)に格納	Aフィールド								
		<table><tr><th>コード</th><th>意 味</th></tr><tr><td>00</td><td>FADD</td></tr><tr><td>01</td><td>FSUB</td></tr></table>	コード	意 味	00	FADD	01	FSUB		
コード	意 味									
00	FADD									
01	FSUB									
第3浮動 小数点演算	FR(N1)×FR(N2)→FR(NT)									
第4浮動 小数点演算	FR(B1)とFR(B2)にBフィールドで示される演算を施してFR(BT)に格納	BフィールドはAフィールドと同じ								
フ ロ ー 制 御	整数演算命令の結果をCCフィールドに従い判定し、条件成立ならPC+dに分岐	Nフィールド								
		<table><tr><th>コード</th><th>意 味</th></tr><tr><td>00</td><td>0: NOP挿入</td></tr><tr><td>01</td><td>1: “ ”</td></tr><tr><td>10</td><td>2: “ ”</td></tr><tr><td>11</td><td>3: “ ”</td></tr></table>	コード	意 味	00	0: NOP挿入	01	1: “ ”	10	2: “ ”
コード	意 味									
00	0: NOP挿入									
01	1: “ ”									
10	2: “ ”									
11	3: “ ”									

【図20】



【図21】

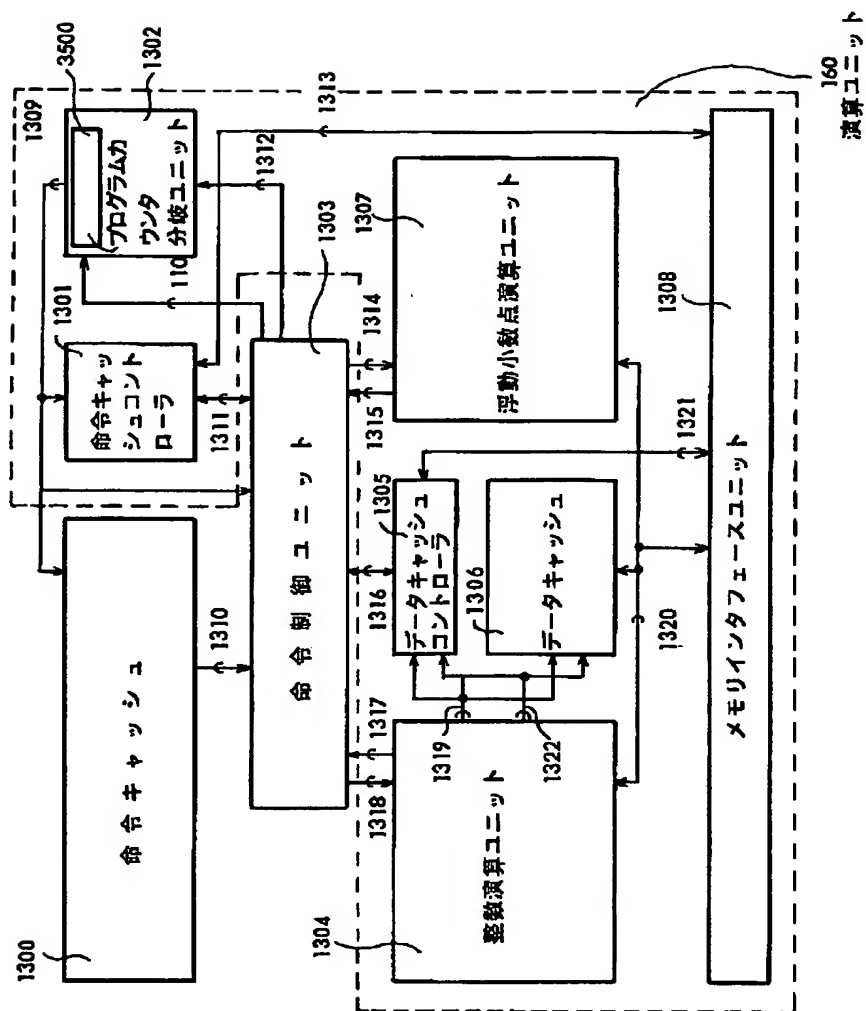


【図12】

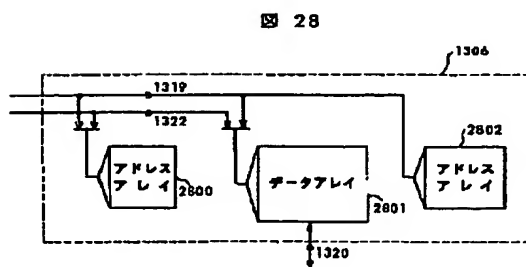
図 12


番号	ロード・ストア演算	乗算減算	第1浮動小数点演算	第2浮動小数点演算	第3浮動小数点演算	第4浮動小数点演算	第5浮動小数点演算	フロート演算
	L: 167-ロード S: 87-ストア X: 何もしない	X: 何もしない	O: FR0+FR31-FR6 Δ: FR2+FR31-FR10 X: 何もしない	O: FR4+FR9-FR12 Δ: FR4+FR10-FR14 X: 何もしない	O: FR1+FR31-FR6 Δ: FR3+FR31-FR11 X: 何もしない	O: FR5+FR9-FR13 Δ: FR2+FR11-FR15 X: 何もしない		X: 何もしない
1	L	X	X	X	X	X	X	X
2	X	X	X	X	X	X	X	X
3	L	X	X	X	X	X	X	X
4	X	X	X	X	X	X	X	X
5	L	X	O	X	O	X	X	X
6	X	X	Δ	X	Δ	X	X	X
7	L	X	O	X	O	X	X	X
8	X	X	Δ	O	Δ	O	X	X
9	L	X	O	Δ	O	Δ	X	X
10	X	X	Δ	O	Δ	O	X	X
11	L	X	O	Δ	O	Δ	X	X
12	S	X	Δ	O	Δ	O	X	X
13	L	X	O	Δ	O	Δ	X	X
14	S	X	Δ	O	Δ	O	X	X
15	L	X	O	Δ	O	Δ	X	X
16	S	X	Δ	O	Δ	O	X	X
17	X	X	X	Δ	X	Δ	X	X
18	S	X	X	O	X	O	X	X
19	X	X	X	Δ	X	Δ	X	X
20	S	X	X	X	X	X	X	X
21	X	X	X	X	X	X	X	X
22	S	X	X	X	X	X	X	X

图 13



【图 3 6】

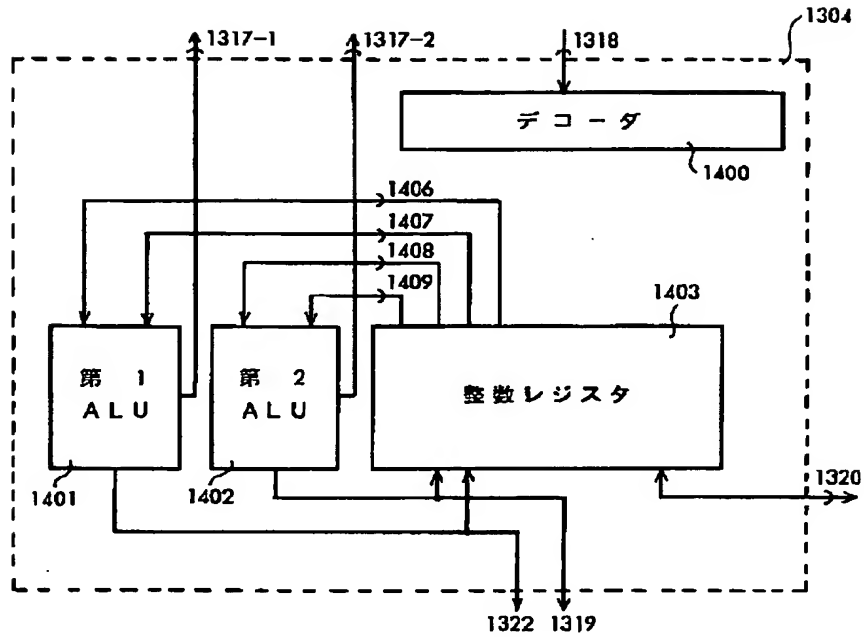


 36

1	5	5	5	5	6	5
1	1	12	1T	J1	J2	JT
6	5	5	2	5	5	5
M1	M?	MT	A	A1	A2	AT
5	5	5	2	5	5	5
N1	N2	NT	B	B1	B2	B3
4	4	4	4	4	10	2
ISZ	I	JSZ	J	CC	d	N

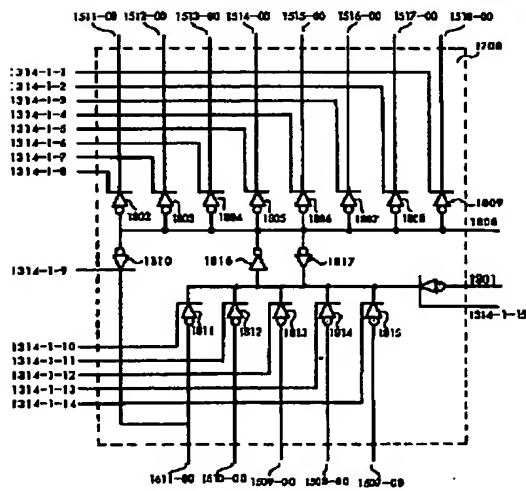
【図14】

図 14



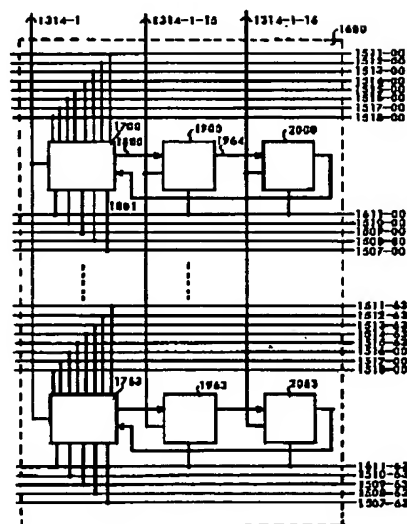
【図18】

図 18



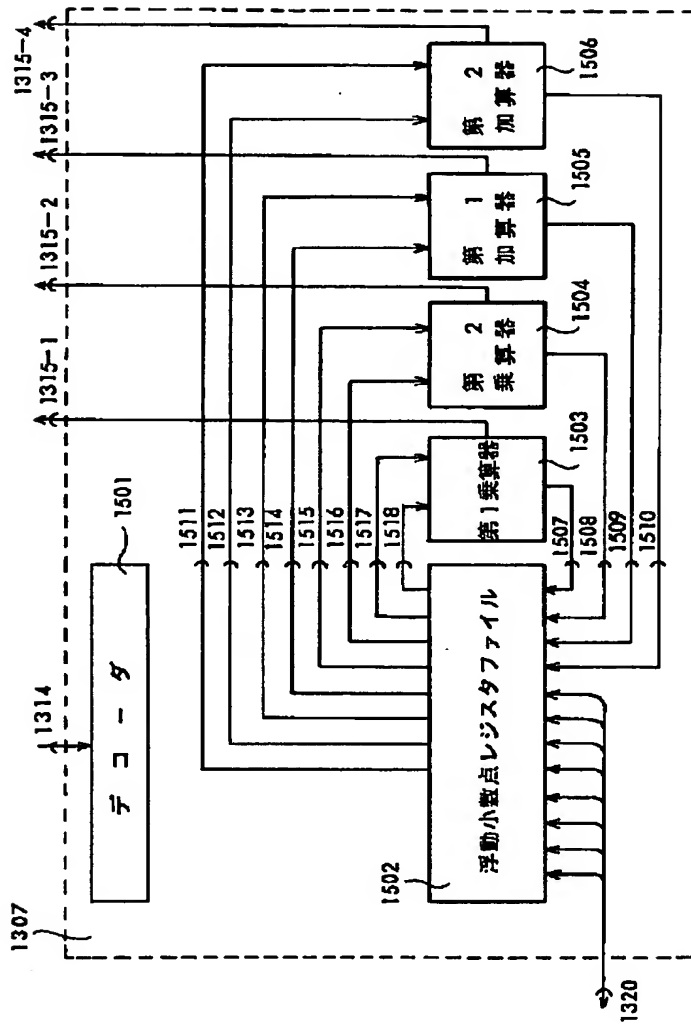
【図19】

図 19



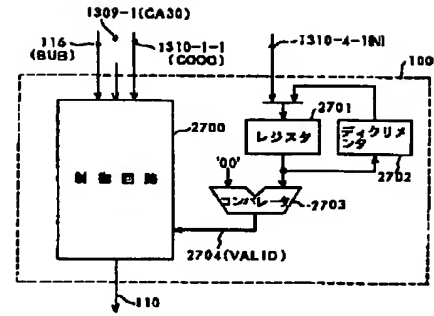
【図15】

図 15



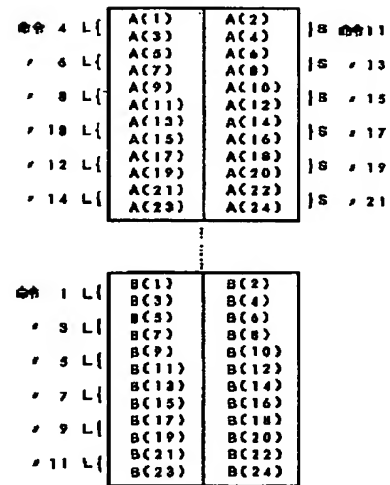
【図27】

図 27



【図38】

図 38



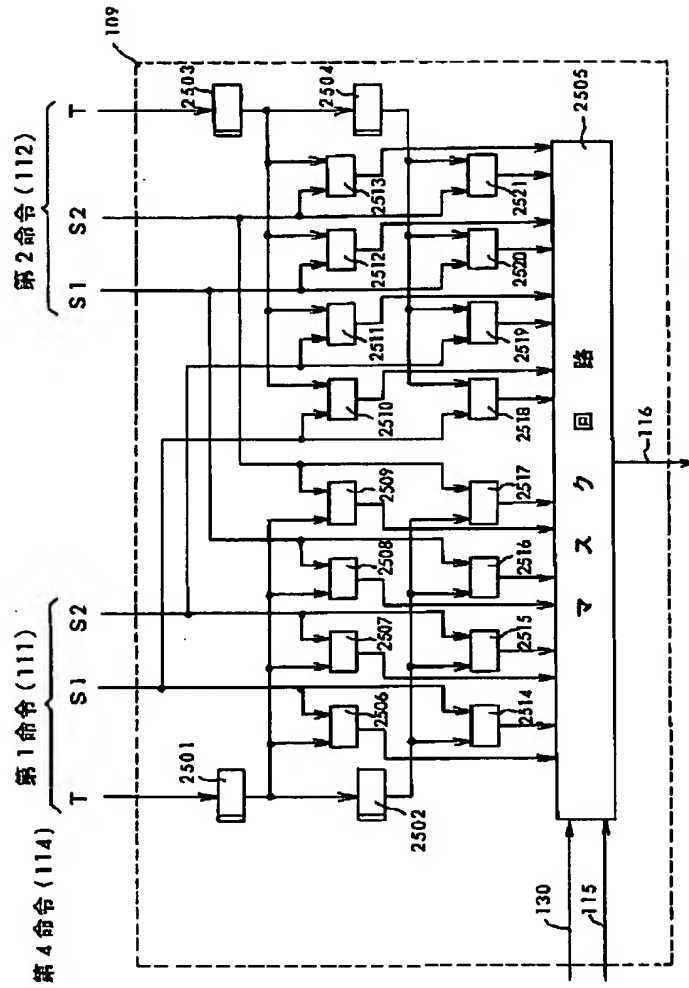
【図24】

図 24

演 算 器	ソース	1 語長命令時	4 語長命令時
第 1 A L U	1	J 1	I 1
	2	J 2	I 2
第 2 A L U	1	A 1	J 1
	2	A 2	J 2
第 1 乗 算 器	1	J 1	M 1
	2	J 2	M 2
第 2 乗 算 器	1	A 1	N 1
	2	A 2	N 2
第 1 加 算 器	1	J 1	A 1
	2	J 2	A 2
第 2 加 算 器	1	A 1	B 1
	2	A 2	B 2

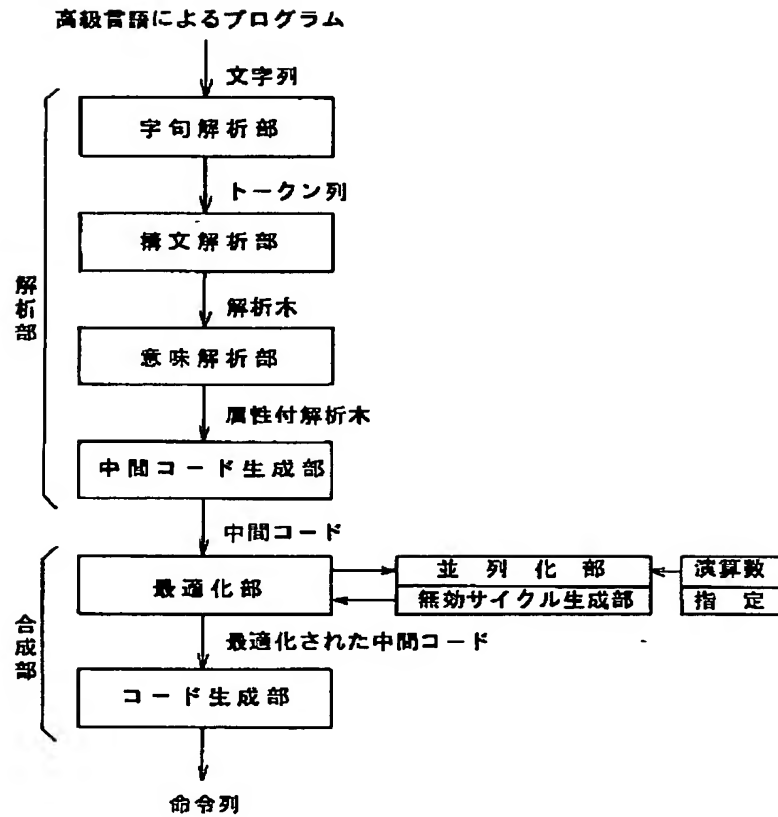
【図25】

図 25



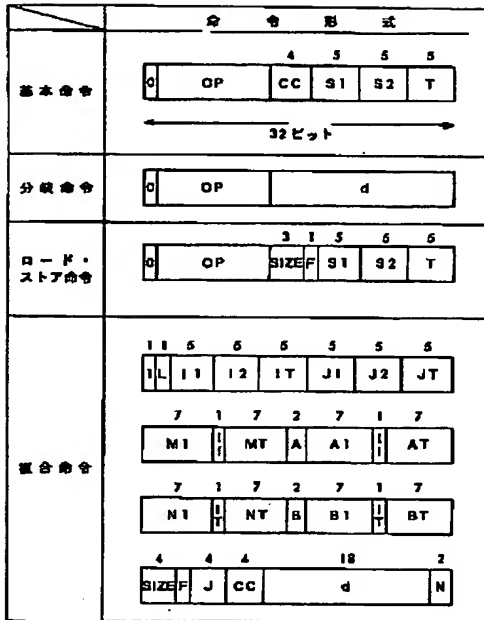
【図26】

図 26



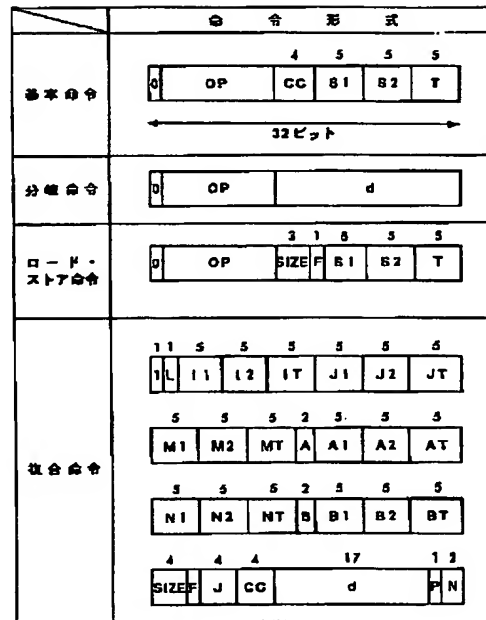
【図31】

図 31



【図33】

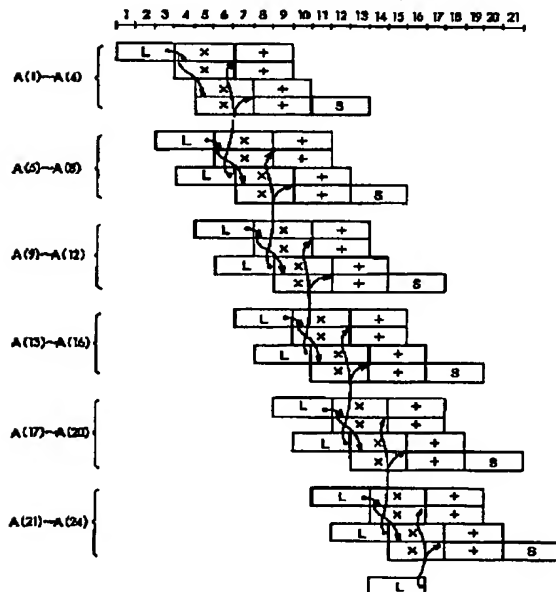
図 33



【図39】

図 39

時 間 (マシナサイクル)



【図32】

図 32

	動 作	補 足										
ロード・ストア演算	I1, I2, SIZE, F, ITフィールドを基本命令のS1, S2, SIZ, F, Tフィールドをみなして、ロード、ストアを実行	L=1…ロード, L=0…ストア, F=1…FR, F=0…R										
整数演算	J≠'1111'の時 R(J1)とR(J2)にJフィールドで示される演算を施して、R(JT)に格納	Jフィールド 0000 ADD, 0001 SUB 0010 AND,										
	J≠'1111'の時 R(J1)+R(J2)で与えられるデータが、キャッシュに無ければ、メモリからキャッシュに転送	JTフィールド 転送するデータの路数を示す。										
第1浮動小数点演算	FR(M1)×FR(MT)→FR(MT)											
第2浮動小数点演算	FR(A1)とFR(AT)にAフィールドで示される演算を施してFR(AT)に格納	Aフィールド <table><tr><th>コード</th><th>意 味</th></tr><tr><td>00</td><td>FADD</td></tr><tr><td>01</td><td>FSUB</td></tr></table>	コード	意 味	00	FADD	01	FSUB				
コード	意 味											
00	FADD											
01	FSUB											
第3浮動小数点演算	NR(N1)×FR(NT)→FR(NT)											
第4浮動小数点演算	FR(B1)とFR(BT)に3フィールドで示される演算を施してFR(BT)に格納	BフィールドはAフィールドと同じ										
フロー制御	整数演算命令の結果をCCフィールドに従い判定し、条件成立ならPC+dに分岐	Nフィールド <table><tr><th>コード</th><th>意 味</th></tr><tr><td>00</td><td>0: NOP 挿入</td></tr><tr><td>01</td><td>1: "</td></tr><tr><td>10</td><td>2: "</td></tr><tr><td>11</td><td>3: "</td></tr></table>	コード	意 味	00	0: NOP 挿入	01	1: "	10	2: "	11	3: "
コード	意 味											
00	0: NOP 挿入											
01	1: "											
10	2: "											
11	3: "											

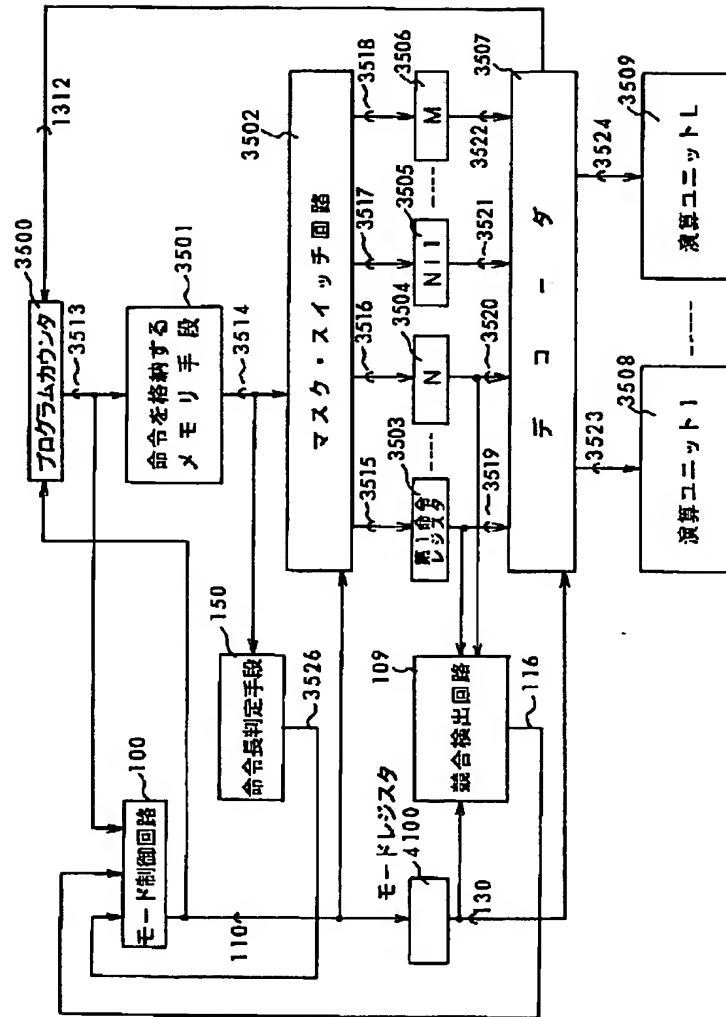
【図34】

図 34

	動 作	補 足										
ロード・ストア演算	I1, I2, S1ZE, F, ITフィールドを基本命令のS1, S2, SIZ, F, Tフィールドをみなして、ロード、ストアを実行	L=1…ロード, L=0…ストア F=1…FR, F=0…R P=1…次のラインをプリフェッチ										
整数演算	J≠'1111'の時 R(J1)とR(J2)にJフィールドで示される演算を施して、R(JT)に格納	Jフィールド 0000 ADD, 0001 SUB 0010 AND,										
第1浮動小数点演算	FR(M1)×FR(M2)→FR(MT)	Aフィールド <table><tr><th>コード</th><th>意 味</th></tr><tr><td>00</td><td>FADD</td></tr><tr><td>01</td><td>FSUB</td></tr></table>	コード	意 味	00	FADD	01	FSUB				
コード	意 味											
00	FADD											
01	FSUB											
第2浮動小数点演算	FR(A1)とFR(A2)にAフィールドで示される演算を施してFR(AT)に格納											
第3浮動小数点演算	FR(N1)×FR(N2)→FR(NT)	BフィールドはAフィールドと同じ										
第4浮動小数点演算	FR(B1)とFR(B2)にBフィールドで示される演算を施してFR(BT)に格納											
フロー制御	整数演算命令の結果をCCフィールドに従い判定し、条件成立ならPC+dに分岐	Nフィールド <table><tr><th>コード</th><th>意 味</th></tr><tr><td>00</td><td>0: NOP 挿入</td></tr><tr><td>01</td><td>1: /</td></tr><tr><td>10</td><td>2: /</td></tr><tr><td>11</td><td>3: /</td></tr></table>	コード	意 味	00	0: NOP 挿入	01	1: /	10	2: /	11	3: /
コード	意 味											
00	0: NOP 挿入											
01	1: /											
10	2: /											
11	3: /											

【図35】

図 35



【図37】

図 37

フィールド	動作	補 足														
第 1 整数演算 I1, I2, IT, ISZ, I	R(I1)とR(I2)を演算して R(IT)に格納	<table><tr><th>I, J</th><th>意 味</th></tr><tr><td>0000</td><td>ロード</td></tr><tr><td>0001</td><td>ストア</td></tr><tr><td>0010</td><td>プリフェッチ</td></tr><tr><td>0011</td><td>ADD</td></tr><tr><td>0100</td><td>SUB</td></tr><tr><td>⋮</td><td>⋮</td></tr></table>	I, J	意 味	0000	ロード	0001	ストア	0010	プリフェッチ	0011	ADD	0100	SUB	⋮	⋮
I, J	意 味															
0000	ロード															
0001	ストア															
0010	プリフェッチ															
0011	ADD															
0100	SUB															
⋮	⋮															
第 2 整数演算 J1, J2, JT, JSZ, J	R(J1)とR(J2)を演算して R(JT)に格納															
第 1 浮動小数点演算 M1, M2, MT	FR(M1)×FR(M2)→FR(MT)															
第 2 浮動小数点演算 A1, A2, AT, A	FR(A1)とFR(A2)を演算して FR(AT)に格納															
第 3 浮動小数点演算 N1, N2, NT	FR(N1)×FR(N2)→FR(NT)	<table><tr><th>ISZ, JSZ</th><th>意 味</th></tr><tr><td>0000</td><td>1 語</td></tr><tr><td>⋮</td><td>⋮</td></tr><tr><td>0111</td><td>256 語</td></tr><tr><td>1000</td><td>1 語</td></tr><tr><td>⋮</td><td>⋮</td></tr><tr><td>1111</td><td>256 語</td></tr></table>	ISZ, JSZ	意 味	0000	1 語	⋮	⋮	0111	256 語	1000	1 語	⋮	⋮	1111	256 語
ISZ, JSZ	意 味															
0000	1 語															
⋮	⋮															
0111	256 語															
1000	1 語															
⋮	⋮															
1111	256 語															
第 4 浮動小数点演算 B1, B2, BT, B	FR(B1)とFR(B2)を演算して FR(BT)に格納															
フロー制御 CC, d	整数演算命令の結果をCCフィールドに従い判定し、条件成立なら PC+dに分岐															
NOP制御 N	N個NOPを挿入	<table><tr><th>A, B</th><th>意 味</th></tr><tr><td>00</td><td>FADD</td></tr><tr><td>01</td><td>FSUB</td></tr></table>	A, B	意 味	00	FADD	01	FSUB								
A, B	意 味															
00	FADD															
01	FSUB															

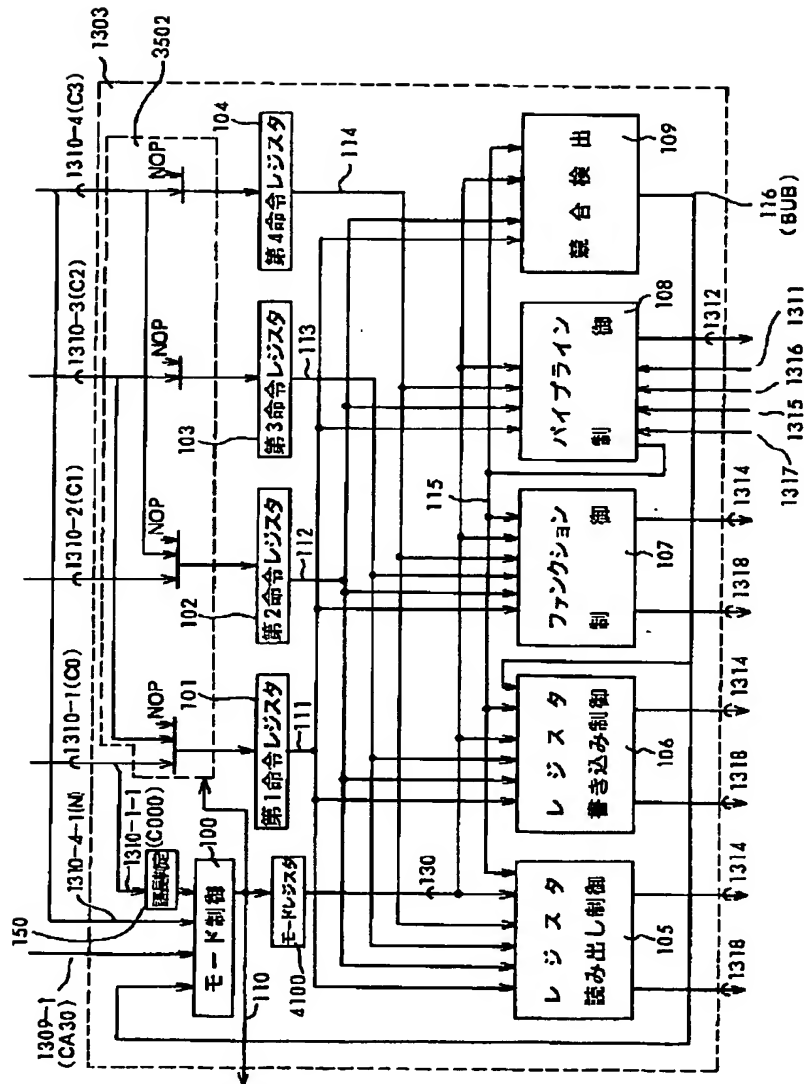
【図40】

図 40

命令	第1段階演算	第2段階演算	第1浮動小数点演算	第2浮動小数点演算	第3浮動小数点演算	第4浮動小数点演算	フロート値
命令1	O: FR0-3△P F Δ: FR4-7△B-F x: 何もしない	O: FR12-15△217 Δ: FR3△R31-FR10 x: 何もしない	O: FR0△R31-FR8 Δ: FR3△R31-FR10 x: 何もしない	O: FR4△R8-FR12 Δ: FR8△R10-FR14 x: 何もしない	O: FR12△R31-FR9 Δ: FR3△R31-FR11 x: 何もしない	O: FR5△R9-FR13 Δ: FR7△R11-FR15 x: 何もしない	x
命令2	x	x	x	x	x	x	x
命令3	x	x	x	x	x	x	x
命令4	Δ	x	Δ	x	Δ	x	x
命令5	Δ	x	Δ	x	Δ	x	x
命令6	Δ	x	Δ	x	Δ	x	x
命令7	Δ	x	Δ	x	Δ	x	x
命令8	Δ	x	Δ	x	Δ	x	x
命令9	Δ	x	Δ	x	Δ	x	x
命令10	Δ	x	Δ	x	Δ	x	x
命令11	Δ	x	Δ	x	Δ	x	x
命令12	Δ	x	Δ	x	Δ	x	x
命令13	x	Δ	Δ	Δ	Δ	Δ	x
命令14	Δ	x	Δ	Δ	Δ	Δ	x
命令15	x	Δ	Δ	Δ	Δ	Δ	x
命令16	x	x	x	Δ	x	Δ	x
命令17	x	Δ	x	Δ	x	Δ	x
命令18	x	x	x	Δ	x	Δ	x
命令19	x	Δ	x	x	x	x	x
命令20	x	x	x	x	x	x	x
命令21	x	Δ	x	x	x	x	x

【図41】

図 41



フロントページの続き

(72)発明者 山田 弘道

茨城県日立市久慈町4026番地 株式会社日立製作所日立研究所内

(72)発明者 前島 英雄

茨城県日立市久慈町4026番地 株式会社日立製作所日立研究所内